BEST AVAILABLE COPY

10/537876

PCT/ 3/05837

Europäisches Patentamt | **European Patent Office** | **Office européen des brevets**

15 DEC 2003

07 JUN 2005

REC'D 22 DEC 2003
WIPO PCT

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

02080613.9

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 02080613.9
Demande no:

Anmeldetag:
Date of filing: 10.12.02
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

Koninklijke Philips Electronics N.V.
Groenewoudseweg 1
5621 BA Eindhoven
PAYS-BAS

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Recording/reproduction and/or editi of real time information on/from a disc like record carrier

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/ Classification internationale des brevets:

G11B27/00

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SI SK

Recording/reproduction and/or editing of real time information on/from a disc like record carrier

The invention relates to an apparatus for recording a real time information signal, such as a digital video signal, on a disc like record carrier, to an apparatus for editing an information signal recorded earlier on said disc like record carrier, to corresponding methods for recording/editing information, to a reading apparatus for reading the information signal and to a record carrier. The record carrier may be of the magnetic or the optical type. An apparatus for recording a real time information signal, such as an MPEG encoded video information signal, on a record carrier is known from WO99/48096 (PHN 17.350). The record carrier in the said document is a disc like record carrier.

Background art.

The layered structure used in BD-RE. This is explained with fig 2-1 on page 6 and fig 2.2 on p7.

Here the definitions are given of:

- The Clip AV stream file,
- The Bridge Clip AV stream file,
- The Clip Information file,
- The PlayList

Clip AV-stream file

The characteristics of Clip AV stream file are given on page 117 (definition of Aligned units and source packets. The addressing in the file is based on source packets.

The definitions of ATC sequences and STC sequences are given on page 65-67.

Clip Information file,

Each Clip AV stream file has a corresponding Clip information file. (p57).

The Clip Information file has some sub-tables. Sub tables important for this Id are: ClipInfo, SequenceInfo and CPI.

PlayList

The PlayList contains a number of PlayItems.

The pointers in the PlayList layer are based on time axis.

The pointers (addresses) in the Clip AV stream file are based on SPN (Source packet numbers).

With the ClipInfo the timing is converted in location in the file (CPI is used for this).

The PlayLists are presented in the Table of Contenst as Titles.

During Playback

A PlayList is selected.

The PlayItems are investigated. IN/OUT time are transferred into SPN of the CLIP AV stream and the source packets which are needed are read from the disc.

Bridges are not used in Real-Playlists but only in Virtual Playlists.

In the apparatuses according to the background art, there is no information how much data is copied to the Bridge clip. For a seamless connection only those source packets which are needed should be read and nothing more. There must be an address (based on source packets numbering where the first PlayItem should be left and where the second PlayItem must be entered. See also fig 4-10.

These information cannot be in the PlayList structure, here only timing is applied.

The invention aims at providing measures to enable a seamless connection while maintaining the PlayList structure which only applies timing information.

The ClipInfo from a Bridge-clip according to the invention contains the SPN of the last Source packet which has to be read in the previous PlayItem and it contains the SPN where the reading of the current PlayItem should start.

Now the procedure is as follows:

The PlayList is selected.

The PlayItems are investigated. If there is a connection=3 between two PlayItems then it is known that the connection is realised with a bridge clip. So there is a reference to the bridgeclip name. table 4.3.4.2

The ClipInfo of this bridge clip has the SPN-exit from preceding clip and the SPN-enter to following clip (table 4.4.2.2).

These and other aspects of the invention will be apparent from and elucidated with reference to the embodiments hereafter in the figure description, in which

Figure 1 shows an embodiment of the apparatus,

Figure 2 shows the recording of blocks of information in fragment areas on the record carrier,

Figure 3 shows the principle of playback of a video information signal,

Figure 4 shows the principle of editing of video information signals,

Figure 5 shows the principle of 'simultaneous' play back and recording,

Figure 6 shows a situation during editing when the generation and recording of a bridging block of information is not required,

Figure 7 shows an example of the editing of a video information signal and the generation of a bridging block of information, at the location of an exit point from the information signal,

Figure 8 shows another example of the editing of a video information signal and the generation of a bridging block of information, at the same location of the exit point as in figure 7,

Figure 9 shows an example of the editing of a video information signal and the generation of a bridging block of information, at the location of an entry point to the information signal,

Figure 10 shows an example of the editing of two information signals and the generation of a bridging block of information,

Figure 11 shows an example of the editing of two information signals and the generation of a bridging block of information, where the editing includes re-encoding some of the information of the two information signals,

Figure 12 shows a further elaboration of the apparatus,

Figure 1 shows an embodiment of the apparatus in accordance with the invention. In the following figure description, the attention will be focussed on the recording, reproduction and editing of a video information signal. It should however be noted that other types of signal could equally well be processed, such as audio signals, or data signals.

The apparatus comprises an input terminal 1 for receiving a video information signal to be recorded on the disc like record carrier 3. Further, the apparatus comprises an

output terminal 2 for supplying a video information signal reproduced from the record carrier 3. The record carrier 3 is a disc like record carrier of the magnetic or optical form.

The data area of the disc like record carrier 3 consists of a contiguous range of physical sectors, having corresponding sector addresses. This address space is divided into fragment areas. A fragment area is a contiguous sequence of sectors, with a fixed length. Preferably, this length corresponds to an integer number of ECC-blocks included in the video information signal to be recorded.

The apparatus shown in figure 1 is shown decomposed into two major system parts, namely the disc subsystem 6 and the what is called 'video recorder subsystem' 8. The following features characterize the two subsystems:

- The disc subsystem can be addressed transparently in terms of logical addresses. It handles defect management (involving the mapping of logical addresses onto physical addresses) autonomously.
- For real-time data, the disc subsystem is addressed on a fragment-related basis. For data addressed in this manner the disc subsystem can guarantee a maximum sustainable bit rate for reading and/or writing. In the case of simultaneous reading and writing, the disc subsystem handles the read/write scheduling and the associated buffering of stream data from the independent read and write channels.
- For non-real-time data, the disc subsystem may be addressed on a sector basis. For data addressed in this manner the disc subsystem cannot guarantee any sustainable bit rate for reading or writing.
- The video recorder subsystem takes care of the video application, as well as file system management. Hence, the disc subsystem does not interpret any of the data that is recorded in the data area of the disc.

In order to realize real time reproduction in all situations, the fragment areas introduced earlier need to have a specific size. Also in a situation where simultaneous recording and reproduction takes place, reproduction should be uninterrupted. In the present example, the fragment size is chosen to satisfy the following requirement:

$$\text{fragment size} = 4\ \text{MB} = 2^{22}\ \text{bytes}$$

Recording of a video information signal will briefly be discussed hereafter, with reference to figure 2. In the video recorder subsystem, the video information signal, which is a real time signal, is converted into a real time file, as shown in figure 2a. A real-

time file consists of a sequence of signal blocks of information recorded in corresponding fragment areas. There is no constraint on the location of the fragment areas on the disc and, hence, any two consecutive fragment areas comprising portions of information of the information signal recorded may be anywhere in the logical address space, as shown in figure 2b. Within each fragment area, real-time data is allocated contiguously. Each real-time file represents a single AV stream. The data of the AV stream is obtained by concatenating the fragment data in the order of the file sequence.

Next, playback of a video information signal recorded on the record carrier will be briefly discussed hereafter, with reference to figure 3. Playback of a video information signal recorded on the record carrier is controlled by means of a what is called 'playback-control-program' (PBC program). In general, each PBC program defines a (new) playback sequence. This is a sequence of fragment areas with, for each fragment area, a specification of a data segment that has to be read from that fragment. Reference is made in this respect to figure 3, where playback is shown of only a portion of the first three fragment areas in the sequence of fragment areas in figure 3. A segment may be a complete fragment area, but in general it will be just a part of the fragment area. (The latter usually occurs around the transition from some part of an original recording to the next part of the same or another recording, as a result of editing.)

Note, that simple linear playback of an original recording can be considered as a special case of a PBC program: in this case the playback sequence is defined as the sequence of fragment areas in the real-time file, where each segment is a complete fragment area except, probably, for the segment in the last fragment area of the file. For the fragment areas in a playback sequence, there is no constraint on the location of the fragment areas and, hence, any two consecutive fragment areas may be anywhere in the logical address space.

Next, editing of one or more video information signals recorded on the record carrier will be briefly discussed hereafter, with reference to figure 4. Figure 4 shows two video information signals recorded earlier on the record carrier 3, indicated by two sequences of fragments named 'file A' and 'file B'. For realizing an edited version of one or more video information signals recorded earlier, a new PBC program should be realized for defining the edited AV sequence. This new PBC program thus defines a new AV sequence obtained by concatenating parts from earlier AV recordings in a new order. The parts may be from the same recording or from different recordings. In order to play back a PBC program, data from various parts of (one or more) real-time files has to be delivered to a decoder. This implies a new data stream that is obtained by concatenating parts of the streams represented by each

~~real-time file. In the figure 4, this is illustrated for a PBC program that uses three parts, one~~ from the file A and two from the file B.

Figure 4 shows that the edited version starts at a point P_1 in the fragment area f(i) in the sequence of fragment areas of figure A and continues until point P_2 in the new fragment area f(i+1) of file A. Then reproduction jumps over to the point P_3 in the fragment area f(j) in file B and continues until point P_4 in fragment area f(j+2) in file B. Next reproduction jumps over to the point P_5 in the same file B, which may be a point earlier in the sequence of fragment areas of file B than the point P_3, or a point later in the sequence than the point P_4.

Next, a condition for seamless playback during simultaneous recording will be discussed. In general, seamless playback of PBC programs can only be realized under certain conditions. The most severe condition is required to guarantee seamless playback while simultaneous recording is performed. One simple condition for this purpose will be introduced. It is a constraint on the length of the data segments that occur in the playback sequences, as follows: In order to guarantee seamless simultaneous play of a PBC program, the playback sequence defined by the PBC program shall be such that the segment length in all fragments (except the first and the last fragment area) shall satisfy:

$$2\ \text{MB} \leq \text{segment length} \leq 4\ \text{MB}$$

The use of fragment areas allows one to consider worst-case performance requirements in terms of fragment areas and segments (the signal blocks stored in the fragment ares) only, as will be described hereafter. This is based on the fact that single logical fragments areas, and hence data segments within fragment areas, are guaranteed to be physically contiguous on the disc, even after remapping because of defects. Between fragment areas, however, there is no such guarantee: logically consecutive fragment areas may be arbitrarily far away on the disc. As a result of this, the analysis of performance requirements concentrates on the following:

a. For playback, a data stream is considered that is read from a sequence of segments on the disc. Each segment is contiguous and has an arbitrary length between 2 MB and 4 MB, but the segments have arbitrary locations on the disc.
b. For recording, a data stream is considered that is to be written into a sequence of 4 MB fragment areas on the disc. The fragment areas have arbitrary locations on the disc.

Note that for playback, the segment length is flexible. This corresponds to the segment condition for seamless play during simultaneous record. For record, however, complete segments areas with fixed length are written.

Given a data stream for record and playback, we will concentrate on the disc subsystem during simultaneous record and playback. It is assumed that the video recorder subsystem delivers a sequence of segment addresses for both the record and the playback stream well in advance.

For simultaneous recording and playback, the disc subsystem has to be able to interleave read and write actions such that the record and playback channels can guarantee sustained performance at the peak rate without buffer overflow or underflow. In general, different R/W scheduling algorithms may be used to achieve this. There are, however, strong reasons to do scheduling in such a way that the R/W cycle time at peak rates is as short as possible:

- Shorter cycle times imply smaller buffer sizes for the read and write buffer, and hence for the total memory in the disc subsystem.
- Shorter cycle times imply shorter response times to user actions. As an example of response time consider a situation where the user is doing simultaneous recording and playback and suddenly wants to start playback from a new position. In order to keep the overall apparatus response time (visible to the user on his screen) as short as possible, it is important that the disc subsystem is able to start delivering stream data from the new position as soon as possible. Of course, this must be done in such a way that, once delivery has started, seamless playback at peak rate is guaranteed. Also, writing must continue uninterruptedly with guaranteed performance.

For the analysis here, a scheduling approach is assumed, based on a cycle in which one complete fragment area is written. For the analysis of drive parameters below, it is sufficient to consider the minimum cycle time under worst-case conditions. Such a worst-case cycle consists of a writing interval in which a 4 MB segment is written, and a reading interval in which at least 4 MB is read, divided over one or more segments. The cycle includes at least two jumps (to and from the writing location), and possibly more, because the segment lengths for reading are flexible and may be smaller than 4 MB. This may result in additional jumps from one read segment location to another. However, since read segments are no smaller than 2 MB, no more than two additional jumps are needed to collect a total of 4 MB. So, a worst-case R/W cycle has a total of four jumps, as illustrated in figure 5. In this figure, x denotes the last part of a read segment, y denoted a complete read segment, with

~~length between 2 MB and 4 MB, and z denotes the first part of a read segment and the total~~ size of x, y and z is again 4 MB in the present example.

In general, the required drive parameters to achieve a guaranteed performance for simultaneous recording and playback depend on major design decisions such as the rotational mode etc. These decisions in turn depend on the media characteristics.

The above formulated conditions for seamless play during simultaneous record are derived such that they can be met by different designs with realistic parameters. In order to show this, we discuss the example of a CLV (constant linear velocity) drive design here.

In the case of a CLV design, transfer rates for reading and writing are the same and independent of the physical location on the disc. Therefore, the worst-case cycle described above can be analyzed in terms of just two drive parameters: the transfer rate R and the worst-case all-in access time τ. The worst-case access time τ is the maximum time between the end of data transfer on one location and the begin of data transfer on another location, for any pair of locations in the data area of the disc. This time covers speed-up/down of the disc, rotational latency, possible retries etc., but not processing delays etc.

For the worst-case cycle described in the previous section, all jumps may be worst-case jumps of duration τ. This gives the following expression for the worst-case cycle time:

$$Tmax = 2F/R_t + 4.\tau$$

where F is the fragment size: $F = 4$ MB $= 33.6\ .10^6$ bits.

In order to guarantee sustainable performance at peak user rate R, the following should hold:

$$F \geq R.Tmax$$

This yields:

$$R \leq F/T_{max} = R_t.F/2.(F + 2R_t.\tau)$$

As an example, with $R_t = 35$ Mbps and $\tau = 500$ ms, we would have: $R \leq 8.57$ Mbps.

Next, editing will be further described. Creating a new PBC program or editing an existing PBC program, generally results in a new playback sequence. It is the objective to guarantee that the result is seamlessly playable under all circumstances, even during simultaneous recording. A series of examples will be discussed, where it is assumed that the intention of the user is to make a new AV stream out of one or two existing AV streams. The examples will be discussed in terms of two streams A and B, where the intention of the user is to make a transition from A to B. This is illustrated in figure 6, where a is the intended exit point from stream A and where b is the intended entry point into stream B.

Figure 6a shows the sequence of fragment areas ......, f(i-1), f(i), f(i+1), f(i+2), ... of the stream A and figure 6b shows the sequence of fragment areas ......, f(j-1), f(j), f(j+1), f(j+2), ... of the stream B. The edited video information signal consists of the portion of the stream A preceding the exit point a in fragment area f(i+1), and the portion of the stream B starting from the entry point b in fragment area f(j).

This is a general case that covers all cut-and-paste-like editing, including appending two streams etc. It also covers the special case where A and B are equal. Depending on the relative position of a and b, this special case corresponds to PBC effects like skipping part of a stream or repeating part of a stream.

The discussion of the examples focuses on achieving seamless playability during simultaneous recording. The condition for seamless playability is the segment length condition on the length of the signal blocks of information stored in the fragment areas, that was discussed earlier. It will be shown below that, if streams A and B satisfy the segment length condition, then a new stream can be defined such that it also satisfies the segment length condition. Thus, seamlessly playable streams can be edited into new seamlessly playable streams. Since original recordings are seamlessly playable by construction, this implies that any edited stream will be seamlessly playable. As a result, arbitrarily editing earlier edited streams is also possible. Therefore streams A and B in the discussion need not be original recordings: they can be arbitrary results of earlier virtual editing steps.

In a first example, a simplified assumption will be made about the AV encoding format and the choice of the exit and entry points. It is assumed that the points a and b are such that, from the AV encoding format point of view, it would be possible to make a straightforward transition. In other words, it is assumed that straightforward concatenation of data from stream A (ending at the exit point a) and data from stream B (starting from entry point b) results in a valid stream, as far as the AV encoding format is concerned.

~~The above assumption implies that in principle a new playback sequence can~~ be defined based on the existing segments. However, for seamless playability at the transition from A to B, we have to make sure that all segments satisfy the segment length condition. Let us concentrate on stream A and see how to ensure this. Consider the fragment area of stream A that contains the exit point a. Let s be the segment in this fragment area that ends at point a, see figure 6a.

If l(s), the length of s, is at least 2 MB, then we can use this segment in the new playback sequence and point a is the exit point that should be stored in the PBC program.

However, if l(s) is less than 2 MB, then the resulting segment s does not satisfy the segment length condition. This is shown in figure 7. In this case a new fragment area, the so-called bridging fragment area f' is created. In this fragment area, a bridging segment comprising a copy of s preceded by a copy of some preceding data in stream A, is stored. For this, consider the original segment r that preceded s in stream A, shown in figure 7a. Now, depending on the length of r, the segment stored in fragment area f(i), either all or part of r is copied into the new fragment area f:

If $l(r) + l(s) \leq 4$ MB, then all of r is copied into f, and the original segment r is not used in the new playback sequence, as illustrated in figure 7a. More specifically, the new exit point is the point denoted a', and this new exit point a' is stored in the PBC program, and later on, after having terminated the editing step, recorded on the disc like record carrier. Thus, in response to this PBC program, during playback of the edited video information stream, after having read the information stored in the fragment area f(i-1), the program jumps to the bridging fragment area f', for reproducing the information stored in the bridging fragment area f', and next jumps to the entry point in the video stream B to reproduce the portion of the B stream, as schematically shown in figure 7b.

If $l(r) + l(s) > 4$ MB, then some part p from the end of r is copied into f', where the length of p is such that we have

$$2\text{ MB} \leq l(r) - l(p) \leq 4\text{ MB} \wedge 2\text{ MB} \leq l(p) + l(s) \leq 4\text{ MB}$$

Reference is made to figure 8, where figure 8a shows the original A stream and figure 8b shows the edited stream A with the bridging fragment area f'. In the new playback sequence, only a smaller segment r' in the fragment area f(i) containing r is now used. This new segment r' is a subsegment of r, viz. the first part of r with length $l(r') = l(r)$ -

l(p). Further, a new exit point a' is required, indicating the position where the original stream A should be left, for a jump to the bridging fragment f'. This new exit position should therefore be stored in the PBC program, and stored later on on the disc.

In the example given above, it was discussed how to create a bridging segment (or: bridging block of information) for the fragment area f', in case the last segment in stream A (i.e. s) becomes too short. We will now concentrate on stream B. In stream B, there is a similar situation for the segment that contains the entry point b, see figure 9. Figure 9a shows the original stream B and figure 9b shows the edited stream. Let t be the segment comprising the entry point b. If t becomes too short, a bridging segment g can be created for storage in a corresponding bridging fragment area. Analogous to the situation for the bridging fragment area f', g will consist of a copy of t plus a copy of some more data from stream B. This data is taken from the original segment u that succeeds t in the fragment area f(j+1) in the stream B. Depending on the length of u, either all or a part of u is copied into g. This is analogous to the situation for r described in the earlier example. We will not describe the different cases in detail here, but figure 9b gives the idea by illustrating the analogy of figure 8, where u is split into v and u'. This results in a new entry point b' in the B stream, to be stored in the PBC program and, later on, on the record carrier.

The next example, described with reference to figure 10, shows how a new seamlessly playable sequence can be defined under all circumstances, by creating at most two bridging fragments (f' and g). It can be shown that, in fact, one bridging fragment area is sufficient, even if both s and t are too short. This is achieved if both s and t are copied into a single bridging fragment area (and, if necessary, some preceding data from stream A and/or some succeeding data from stream B). This will not be described extensively here, but figure 10 shows the general result.

In examples described above, it was assumed that concatenation of stream data at the exit and entry points a and b was sufficient to create a valid AV stream. In general, however, some re-encoding has to be done in order to create a valid AV stream. This is usually the case if the exit and entry points are not at GOP boundaries, when the encoded video information signal is an MPEG encoded video information signal. The re-encoding will not be discussed here, but the general result will be that some bridge sequence is needed to go from stream A to stream B. As a consequence, there will be a new exit point a' and a new entry point b', and the bridge sequence will contain re-encoded data that corresponds with the original pictures from a' to a followed by the original pictures from b to b'.

Not all the cases will be described in detail here, but the overall result is like in the previous examples: there will be one or two bridging fragments to cover the transition from A to B. As opposed to the previous examples, the data in the bridging fragments is now a combination of re-encoded data and some data copied from the original segments. Figure 11 gives the general flavour of this.

As a final remark, note that one does not have to put any special constraints on the re-encoded data. The re-encoded stream data simply has to satisfy the same bitrate requirements as the original stream data.

Figure 12 shows a schematic version of the apparatus in more detail. The apparatus comprises a signal processing unit 100 which is incorporated in the subsystem 8 of figure 1. The signal processing unit 100 receives the video information signal via the input terminal 1 and processes the video information into a channel signal for recording the channel signal on the disc like record carrier 3. Further, a read/write unit 102 is available which is incorporated in the disc subsystem 6. The read/write unit 102 comprises a read/write head 104, which is in the present example an optical read/write head for reading/writing the channel signal on/from the record carrier 3. Further, positioning means 106 are present for positioning the head 104 in a radial direction across the record carrier 3. A read/write amplifier 108 is present in order to amplify the signal to be recorded and amplifying the signal read from the record carrier 3. A motor 110 is available for rotating the record carrier 3 in response to a motor control signal supplied by a motor control signal generator unit 112. A microprocessor 114 is present for controlling all the circuits via control lines 116, 118 and 120.

The signal processing unit 110 is adapted to convert the video information received via the input terminal 1 into blocks of information of the channel signal having a specific size. The size of the blocks of information (which is the segment mentioned earlier) can be variable, but the size is such that it satisfies the following relationship:

$$SFA/2 \leq \text{size of a block of the channel signal} \leq SFA,$$

where SFA equals the fixed size of the fragment areas. In the example given above, SFA = 4 MB. The write unit 102 is adapted to write a block of information of the channel signal in a fragment area on the record carrier.

In order to enable editing of video information recorded in an earlier recording step on the record carrier 3, the apparatus is further provided with an input unit 130 for

receiving an exit position in a first video information signal recorded on the record carrier and for receiving an entry position in a second video information signal recorded on that same record carrier. The second information signal may be the same as the first information signal. Further, the apparatus comprises a memory 132, for storing information relating to the said exit and entry positions. Further the apparatus comprises a bridging block generating unit 134, incorporated in the signal processing unit 100, for generating at least one bridging block of information (or bridging segment) of a specific size. As explained above, the bridging block of information comprises information from at least one of the first and second video information signals, which information is located before the exit position in the first video information signal and/or after the entry position in the second video information signal. During editing, as described above, one or more bridging segments are generated in the unit 134 and in the edit step, the one or more bridging segment(s) is (are) recorded on the record carrier 3 in a corresponding fragment. The size of the at least one bridging block of information also satisfies the relationship:

$$SFA/2 \leq \text{size of a bridging block of information} \leq SFA.$$

Further, the PBC programs obtained in the edit step can be stored in a memory incorporated in the microprocessor 114, or in another memory incorporated in the apparatus. The PBC program created in the edit step for the edited video information signal will be recorded on the record carrier, after the editing step has been terminated. In this way, the edited video information signal can be reproduced by a different reproduction apparatus by retrieving the PBC program from the record carrier and reproducing the edited video information signal using the PBC program corresponding to the edited video information signal.

In this way, an edited version can be obtained, without re-recording portions of the first and/or second video information signal, but simply by generating and recording one or more bridging segments into corresponding (bridging) fragment areas on the record carrier.

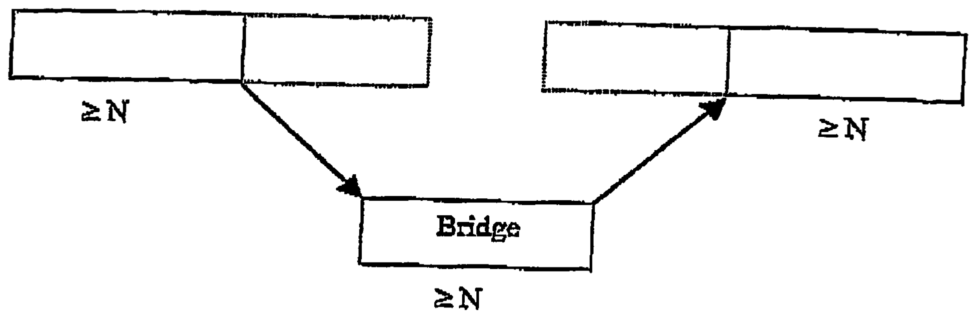
Whilst the invention has been described with reference to preferred embodiments thereof, it is to be understood that these are not limitative examples. Thus, various modifications may become apparent to those skilled in the art, without departing from the scope of the invention, as defined by the claims. The disclosed fragment size of 4 MB is characteristic of one specific embodiment. Another embodiments may use other fragment sizes, such as 6 MB for example.

Further, the invention lies in each and every novel feature or combination of features. The invention can be implemented by means of both hardware and software, and that several "means " may be represented by the same item of hardware. Furthermore, the word "comprising" does not exclude the presence of other elements or steps than those listed in the claims.

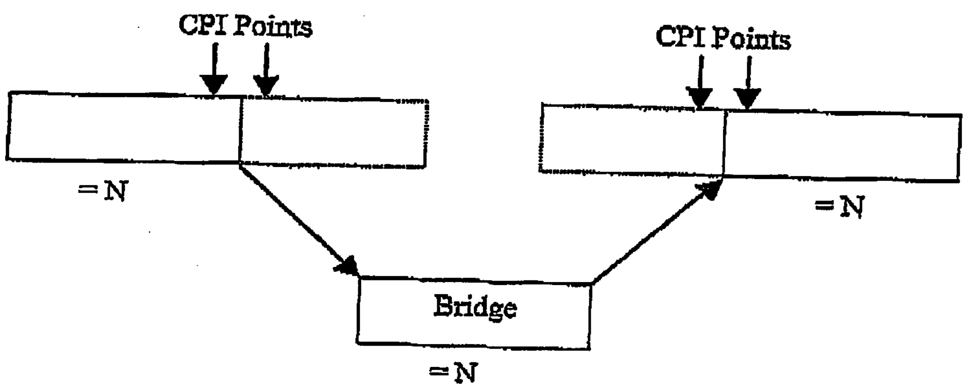
ID605718 / Int ref 27079

In DVR we have an allocation rule that says that each contiguous extent must have a minimum size of N (where in the latest version N = 12 MB).

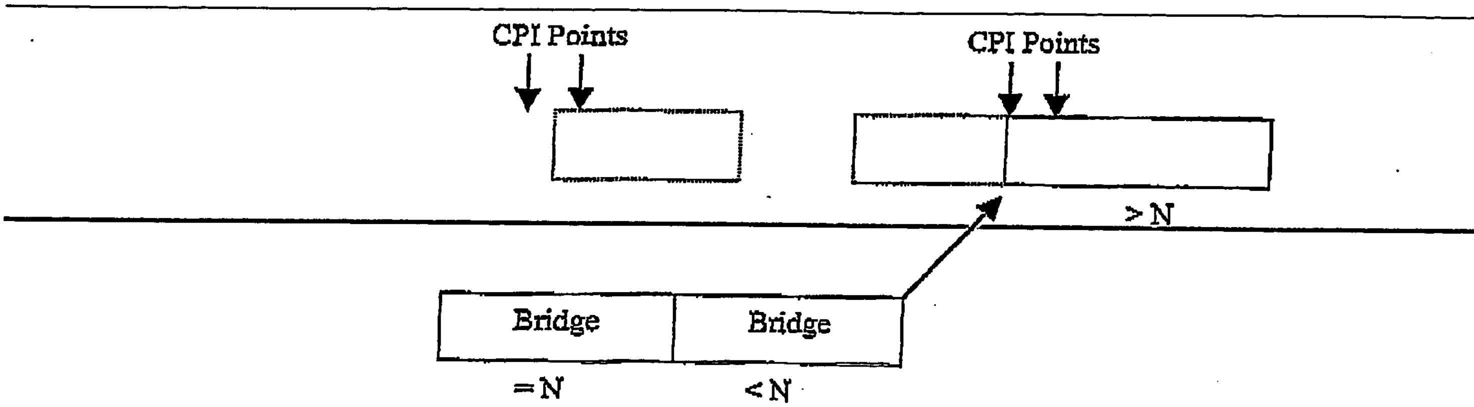
When editing with a bridge sequence, it is necessary to ensure that the extent before the bridge sequence, the bridge sequence itself and the segment after the bridge sequence all satisfy the minimum extent size.



Now suppose you have a borderline case where the three extents all equal (approximately) N e.g.



Now with the current addressing scheme (based on source packet numbers) this is no problem. However, if you address the jump to/from the bridge using time and then using CPI to resolve the time to location, the points in CPI determine where you can jump. This means you will have to either copy more or less data from the original streams to the bridge - either one will violate the allocation rule. This means you will need to copy one of the extents from the original sequence to the bridge which result in the following:



In fact, depending on how you do the allocation, the result could be much worse. If you do allocation in blocks of N then when you are creating the bridge you will need to copy either all of an extent or none of it. However, the CPI locations are based on the video content, not the allocation extents so in general the CPI points will never correspond to the start of an allocation extent. This problem is more severe in the new allocation scheme because the minimum allocation extent size equals the fragment size.

Even with our addressing scheme it may be necessary in some cases to copy more extents to the bridge sequence but by using the packet based addressing we reduce the number of cases to a minimum.

2 Application concepts

2.1 Introduction

This chapter describes basic concepts about the application format of the MPEG-2 transport stream specifications, and explains data structures that are specified in subsequent chapters.

2.2 Format structure

Figure 2-1 below describes a simplified structure of the application format. The application format has two layers for managing AV stream files: those are ***PlayList*** and ***Clip***. The BDAV Information manages all the Clips and the PlayLists in a BDAV directory.

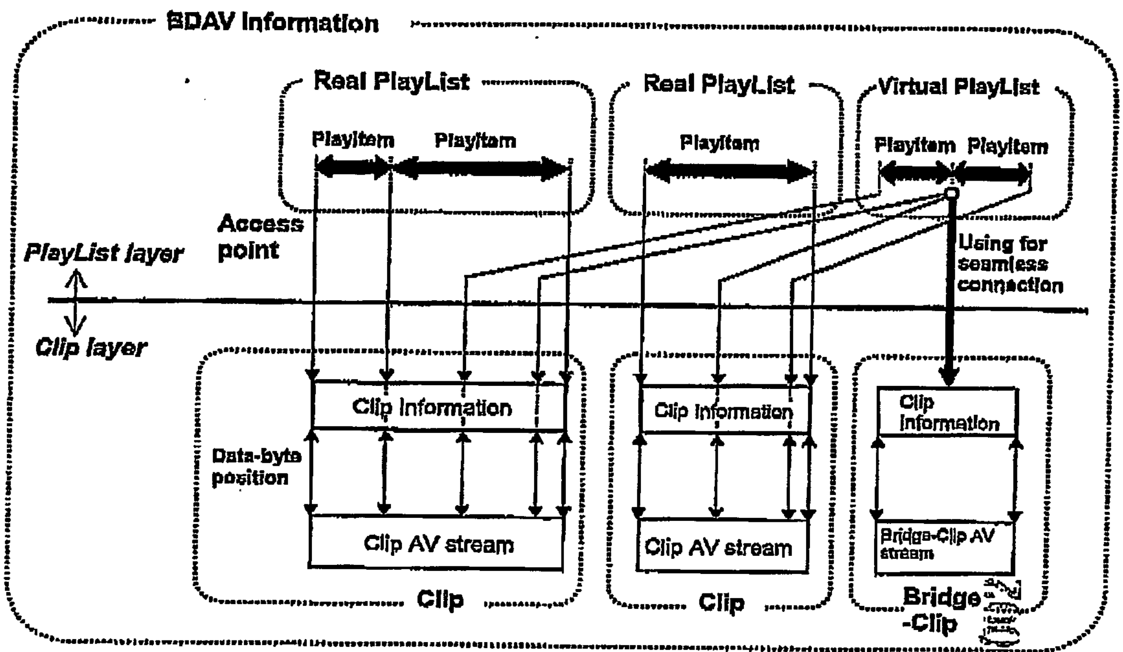


Figure 2-1 – Simplified structure of the application format

2.2.1 Clip

Each pair of an AV stream file and its attribute is considered to be one object. The AV stream file is called a *Clip AV stream file* or a *Bridge-Clip AV stream file*, and the attribute is called a *Clip Information file*.

Each object of a Clip AV stream file and its Clip Information file is called a ***Clip***. The Clips are normal Clips in this specification.

Each object of a Bridge-Clip AV stream file and its Clip Information file is called a ***Bridge-Clip***. The Bridge-Clips are special Clips that are used for special purpose described in the following sections.

(1) AV stream file

(1-1) Clip AV stream file

Clip AV stream files store data that is formatted an MPEG-2 transport stream to a structure defined by this specification. The structure is called the *BDAV MPEG-2 transport stream.*

Clip AV stream files are normal AV stream files in this specification. A Clip AV stream file is created on the BDAV directory, when the recorder encodes analogue input signals to an MPEG-2 transport stream and records the stream or when the recorder records an input digital broadcast stream.

(1-2) Bridge-Clip AV stream file

Bridge-Clip AV stream files also have the BDAV MPEG-2 transport stream structure.

Bridge-Clip AV stream files are special AV stream files that are used for making seamless connection between two presentation intervals selected in the Clips.

Generally, Bridge-Clip AV stream files have very small data size compared to Clip AV stream files.

(2) Clip Information file

In general, a file is regarded as a sequence of data bytes, but the contents of the AV stream file (Clip AV stream or Bridge-Clip AV stream) is developed on a time axis. The access points in the AV stream file are specified mostly with time stamp basis.

When the time stamp of the access point is given to the AV stream file, the Clip Information file finds the addressing information of the position where the player should start to read the data in the AV stream file.

One AV stream file has one associated Clip Information file.

2.2.2 PlayList

PlayList is introduced to be able to edit easily playing intervals in the Clips that the user wants to play, e.g., assemble editing without moving, copying or deleting the part of Clips in the BDAV directory.

A PlayList is a collection of playing intervals in the Clips. Basically, one playing interval is called a *PlayItem* and is a pair of IN-point and OUT-point that point to positions on a time axis of the Clip. Therefore a PlayList is a collection of PlayItems. Here the IN-point means a start point of a playing interval, and the OUT-point means an end point of the playing interval.

There are two types of PlayList: those are a *Real-PlayList* and a *Virtual-PlayList* (See Figure 2-2).

(1) Real-PlayList

A Real-PlayList can use only Clip AV stream files, and can not use Bridge-Clip AV stream files.

The Real-PlayList is considered that it has its referring parts of Clips. So, the Real-PlayList is considered that it occupies the data space that is equivalent to its referring parts of Clips in the disc (the data space is mainly occupied by the AV stream files).

When the Real-PlayList is deleted, the referring parts of Clips are also deleted.

(2) Virtual-PlayList

A Virtual-PlayList can use both Clip AV stream files and Bridge-Clip AV stream files.

The Virtual-PlayList is considered that it does not have the data of Clip AV stream files but it has the data of Bridge-Clip AV stream files if it uses the Bridge-Clip AV stream files.

When the Virtual-PlayList that does not use the Bridge-Clip AV stream files is deleted, the Clips do not change.

When the Virtual-PlayList that uses the Bridge-Clip AV stream files is deleted, the Clip AV stream files and the associated Clip Information files do not change, but the Bridge-Clip AV stream files and the associated Clip Information file used by the Virtual-PlayList are also deleted.

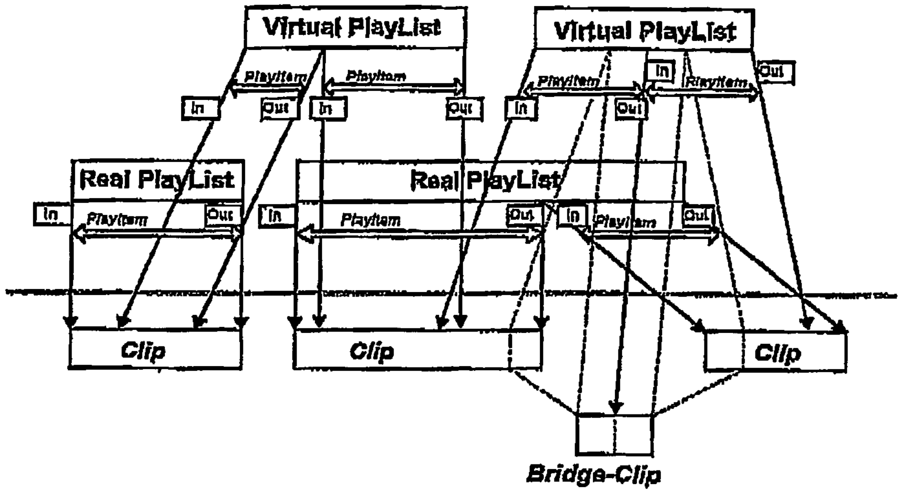


Figure 2-2 – An illustration of Real-PlayList and Virtual-PlayList

2.3 User interface concept

The Clips are only internal to the player/recorder-system and are not visible in the user interface of the player/recorder-system. Only the PlayLists are shown to the user.

2.4 Operation for PlayList

2.4.1 Operation for Real-PlayList

<u>(1) "Create" the Real-PlayList</u>

The Real-PlayList created at the first recording can only have PlayItems that refer to the whole Clip.

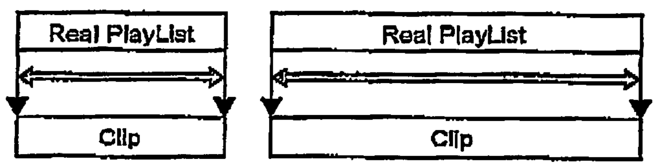


Figure 2-3 – An example of Creating the PlayList

<u>(2) "Divide" the Real-PlayList</u>

Dividing one Real-PlayList into two, and constructing two Real-PlayLists. This operation makes no change in the Clips.

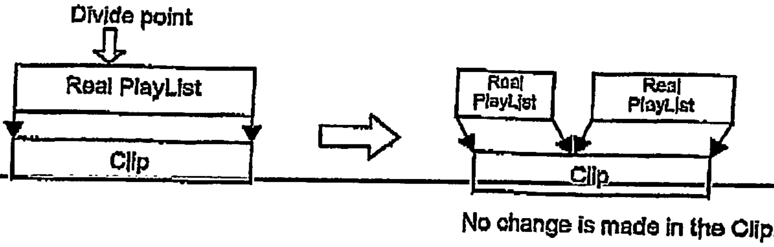


Figure 2-4 -- An example of dividing the Real-PlayList

(3) "Combine" the Real-PlayLists

Combining two Real-PlayLists into one new Real-PlayList. This operation makes no change in the Clips.

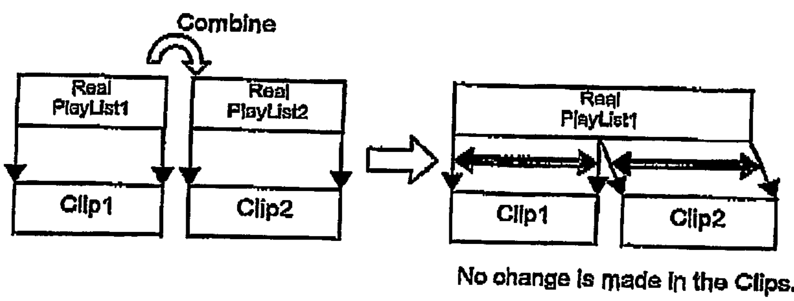


Figure 2-5 -- An example of combining two Real-PlayLists

(4) "Delete" the whole Real-PlayList

Deleting the whole Real-PlayList. The referring parts of Clips are also deleted with this operation.

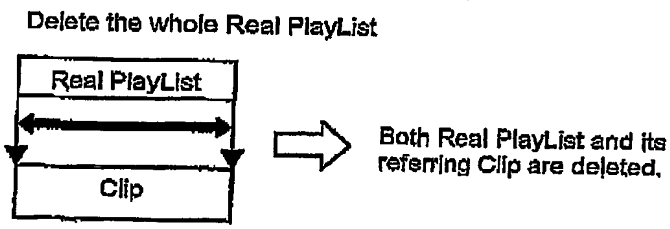


Figure 2-6 -- An example of deleting the whole Real-PlayList

(5) "Delete" the part of the Real-PlayList

Changing PlayItems of the Real-PlayList that makes to use parts of Clips that are still needed. The other parts that are not needed in the Clips are deleted.

Figure 2-7 below illustrates an example of deleting the beginning part of the Real-PlayList.

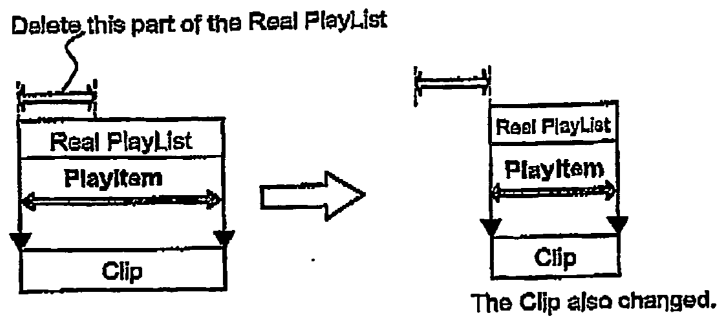


Figure 2-7 -- An example of deleting a beginning part of the Real-PlayList

When the middle of a Clip AV stream is deleted by editing, the parts originated from the same Clip may be combined into a Clip file (see Figure 2-8).

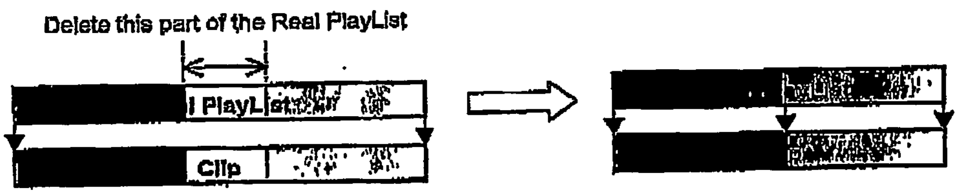


Figure 2-8 -- An example of deleting a middle part of the Real-PlayList

If a Real-PlayList and parts of Clips used by the Real-PlayList are changed, there may be a conflict with Virtual-PlayLists using the same parts of Clips. In this case, the user interface may do the following action.

- Warning and asking for the "delete" operation to the user that if the Real-PlayList and the parts of Clips used by the Real-PlayList are deleted then the Virtual-PlayLists using the same parts of Clips will be also deleted.

or

- The user interface should be suggested to keep the Virtual-PlayList files and just eliminate PlayItems that have lost parts of Clips to be referred to, from the Virtual-PlayList.

2.4.2 Operation for Virtual-PlayList

(1) "Assemble" editing (IN-OUT editing)

Making PlayItems that the user wants to play, then combining the PlayItems into a Virtual-PlayList (See Figure 2-9).

This specification supports to make a seamless presentation through a connection point between two PlayItems by making a Bridge-Clip (See Figure 2-10). Since it is possible to play the MPEG video stream seamlessly at the connection point, normally a small number of pictures around the connection point must be re-encoded, and the Bridge-Clip contains the re-encoded pictures. This operation makes no change in the Clip AV stream files and the associated Clip Information files.

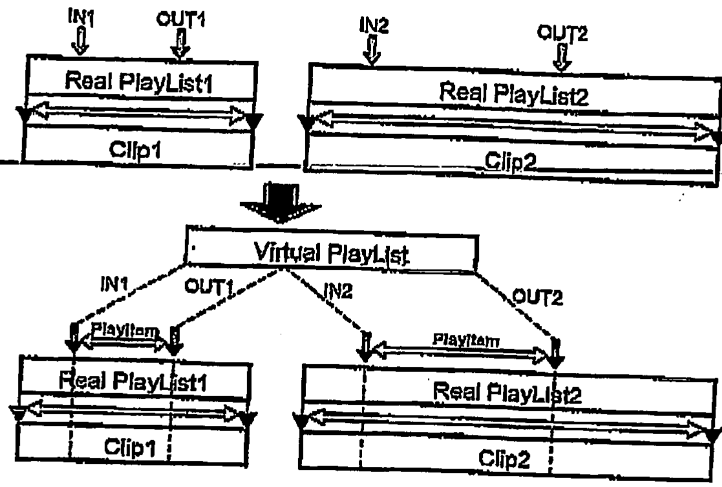


Figure 2-9 – An example of assemble editing (Non-seamless connection between two PlayItems)

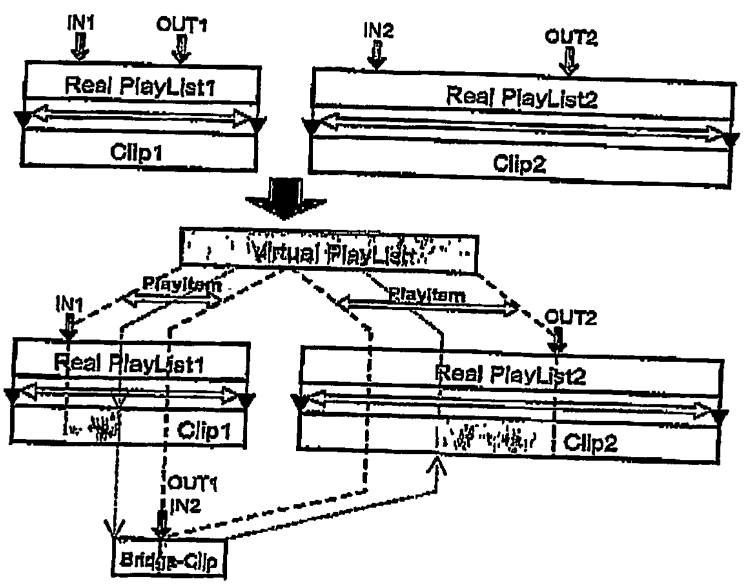


Figure 2-10 – An example of assemble editing (Seamless connection between two PlayItems)

(2) "Re-editing" the Virtual-PlayList

This operation is considered as one of the following actions: Changing the IN-point and/or the OUT-point of the PlayItem in the Virtual-PlayList, appending or inserting a new PlayItem to the Virtual-PlayList, or deleting the PlayItem in the Virtual-PlayList.

If the user will change the IN-point and/or the OUT-point that refers to a Bridge-Clip, the recorder should give a warning and asking for the action to the user that the Bridge-Clip will be deleted and

needs to create a new Bridge-Clip for making a seamless connection. And if the answer is yes, the recorder may delete the old Bridge-Clip and may create the new Bridge-Clip.

(3) "Delete" the Virtual-PlayList

Deleting the whole Virtual-PlayList.

(4) "Audio dubbing (post recording)" to the Virtual-PlayList

This specification supports the audio dubbing (post-recording) function to the Virtual-PlayList. Auxiliary audio streams can be attached as the sub path to the main path of the Virtual-PlayList.

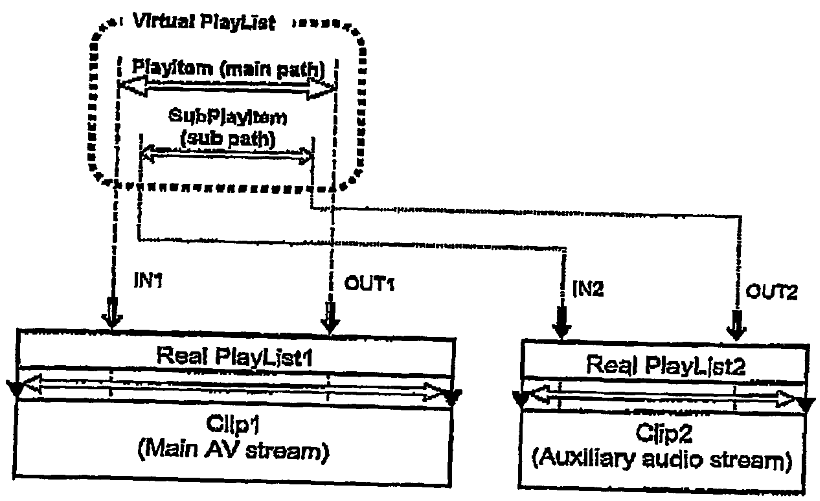


Figure 2-11 -- An example of Audio dubbing operation

2.4.3 Common operation for both Real and Virtual-PlayLists

(1) "Moving" the presentation order of the PlayLists in the BDAV directory

Changing the presentation order of PlayLists in the BDAV directory. This operation is supported by *TableOfPlayLists* defined in 4.2.3. This operation makes no change in the Clips.

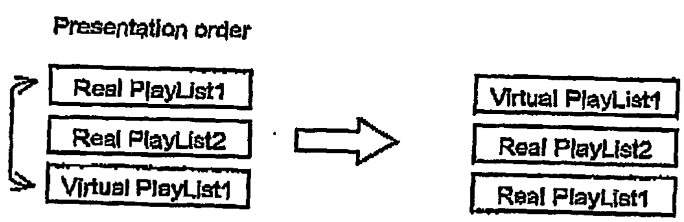


Figure 2-12 -- An example of moving operation

2.5 Mark

Marks are used for specifying the time stamps of highlights or characteristic scenes on Clips, Bridge-Clips and PlayLists.

Each Clip Information file and PlayList file has a data block that stores the list of Marks. Setting a Mark is only adding the time stamp to the list and deleting a Mark is to remove the time stamp from the list. Therefore there is no modification to the AV stream files.

The PlayList can refer to the Marks that are attached to the Clip used by the PlayList.

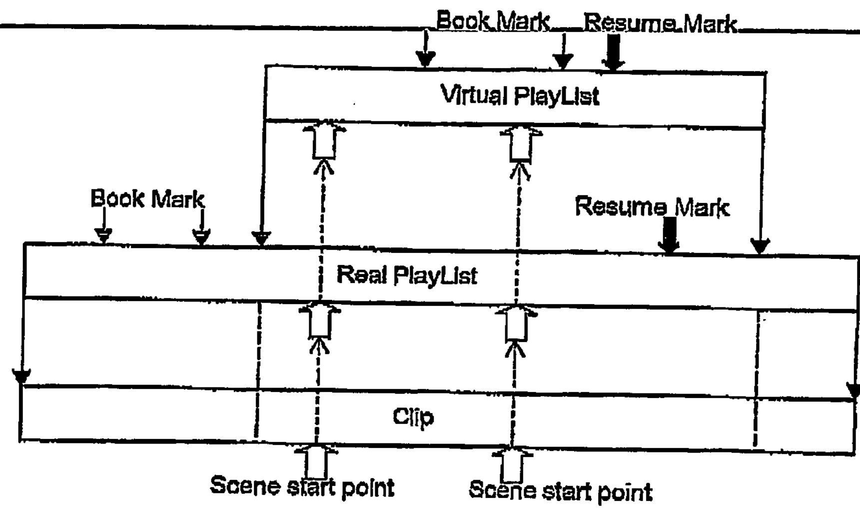


Figure 2-13 – Marks on PlayList and Marks on a Clip

2.6 Thumbnail

Thumbnails are still pictures that are attached to the BDAV directory, PlayLists, Clips and Bridge-Clips. There are two types of thumbnails. One is a representative picture that summarizes the contents. It will be mainly used in the Menu screen on which the user moves a pointer and selects what the user wants to see. The other type is the picture that shows the scene the Mark points to.

Each BDAV directory and each PlayList can have only one representative picture. We call these pictures *Menu Thumbnails*. These Menu Thumbnails are displayed frequently; therefore they are required to be read quickly from the BDAV directory. For this requirement, it is effective to collect all the Menu Thumbnails into one file.

On the other hand, Each PlayList, Clip and Bridge-Clip can have plural Marks and the associated Mark pictures. We call these pictures *Mark Thumbnails*. Unlike the Menu Thumbnails, the Mark Thumbnails will be used in a sub-menu that shows the details of each PlayList, and do not require short access time. Therefore it will not be a serious problem that the player opens and reads a file every time it needs a thumbnail. To reduce the number of files in a BDAV directory, all the Mark Thumbnails for all the PlayLists, Clips and Bridge-Clips in the BDAV directory are stored in one file.

Each PlayList can have both a Menu Thumbnail and Mark Thumbnails, but Each Clip cannot have a Menu Thumbnail, but can have Mark Thumbnails.

Thumbnail	Menu Thumbnail	⟹	0 or 1	per	BDAV directory, PlayList
	Mark Thumbnail	⟹	0 or above	per	PlayList, Clip

4.3.4 PlayItem

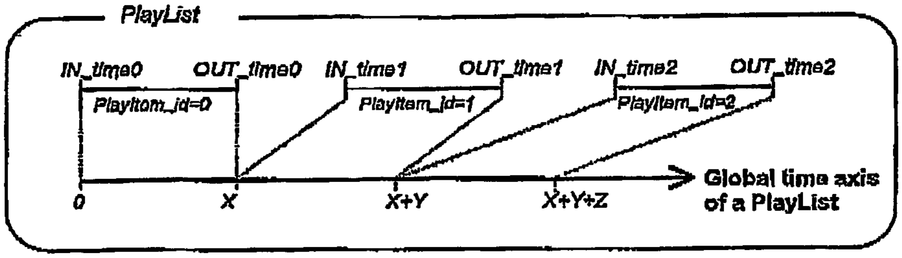
This section defines the syntax and semantics of **PlayItem()** in the PlayList().

4.3.4.1 PlayItem - General

The PlayItem specifies a time based playing interval from the IN_time until the OUT_time. The playing interval basically refers to a Clip, and optionally may refer to a Clip and a Bridge-Clip.

When a PlayList is composed of two or more PlayItems, the playing intervals of these PlayItems shall be placed in line without a time gap or overlap on a *Global time axis of the PlayList* (see Figure 4-4).

The Global time axis may be visible in the user interface on the system, and the user can command a start time of the playback on the global time axis to the system, e.g. the playback is started 30 minutes after the beginning in the PlayList.



X : Length of the playing interval of PlayItem referred by PlayItem_id=0.
Y : Length of the playing interval of PlayItem referred by PlayItem_id=1.
Z : Length of the playing interval of PlayItem referred by PlayItem_id=2.

(The Length of the playing interval of PlayItem is defined in 4.3.4.3.2.)

Figure 4-4 – An example of the global time axis of a PlayList

4.3.4.1.1 Current PlayItem and Previous PlayItem

In the following section, when we consider the connection of two PlayItems, we use the words ***"Current PlayItem" and "Previous PlayItem"***. These two PlayItems appear in the PlayList consecutively, and the previous PlayItem is connected immediately ahead with the current PlayItem (see Figure 4-5).

The "***IN_time of the current PlayItem***" means the IN_time of which the current PlayItem has.

The "***OUT_time of the current PlayItem***" means the OUT_time of which the current PlayItem has.

The "***IN_time of the previous PlayItem***" means the IN_time of which the previous PlayItem has.

The "***OUT_time of the previous PlayItem***" means the OUT_time of which the previous PlayItem has.

Figure 4-5 – Relationship between the *current PlayItem* and the *previous PlayItem*

4.3.4.1.2 connection_condition

When the previous PlayItem and the current PlayItem are connected in the PlayList, the current PlayItem has a connection condition between *the IN_time of the current PlayItem* and *the OUT_time of the previous PlayItem*. The *connection_condition* field of the current PlayItem indicates the connection condition.

4.3.4.1.3 BridgeSequenceInfo

When the previous PlayItem and the current PlayItem are connected with a Bridge-Clip, the current PlayItem has the *BridgeSequenceInfo* (See 4.3.5).

The BridgeSequenceInfo gives a name of Clip Information file to specify a Bridge-Clip AV stream file. And the Clip Information file for the Bridge-Clip AV stream file gives information for the connection between the previous PlayItem and the current PlayItem (See the semantics of *preceding_Clip_Information_file_name, SPN_exit_from_preceding_Clip, following_Clip_Information_file_name* and *SPN_enter_to_following_Clip* defined in the 4.4.2).

4.3.4.2 PlayItem - Syntax

Syntax	No. of bits	Mnemonic
PlayItem() {		
length	16	uimsbf
Clip_Information_file_name	8*5	bslbf
Clip_codec_identifier	8*4	bslbf
reserved_for_future_use	12	bslbf
connection_condition	4	uimsbf
if (*CPI_type*==1) { /* the CPI_type is defined in CPI() of the Clip Information file referred to by the Clip_Information_file_name. */		
ref_to_STC_id	8	uimsbf
} else {		
reserved_for_word_align	8	bslbf
}		
IN_time	32	uimsbf
OUT_time	32	uimsbf
if(<Virtual-PlayList> && *connection_condition*==3){		
BridgeSequenceInfo()		
}		
}		

4.3.4.3 PlayItem - Semantics

length: This 16-bit field indicates the number of bytes of the *PlayItem()* immediately following this *length* field and up to the end of the *PlayItem()*.

Clip_Information_file_name: This field specifies the name of a Clip Information file for the Clip used by the PlayItem. This field shall contain the 5-digit number "zzzzz" of the name of the Clip except the extension. It shall be coded according to ISO 646. The *Clip_stream_type* field in the *ClipInfo* of the Clip Information file shall indicate "a Clip AV stream of the BDAV MPEG-2 transport stream".

Clip_codec_identifier: This field shall have the value "M2TS" coded according to ISO 646 in the PlayList file using BDAV MPEG-2 transport streams.

All PlayItems in a PlayList shall have the same value "M2TS" in the *Clip_codec_identifier*,

If the *PL_CPI_type* in PlayList() is set to 1 and the *Clip_codec_identifier* is "M2TS", each Clip used by the PlayList file (for BDAV MPEG-2 transport streams) shall have an EP_map in the CPI().

If the *PL_CPI_type* in PlayList() is set to 2 and the *Clip_codec_identifier* is "M2TS", each Clip used by the PlayList file (for BDAV MPEG-2 transport streams) shall have a TU_map in the CPI().

connection_condition: This 4-bit field indicates the connection condition between the *IN_time* of the current PlayItem and the *OUT_time* of the previous PlayItem. Only values 1 to 4 are permitted for the connection_condition.

If the PlayItem is the first PlayItem in the PlayList, the connection_condition has no meaning and shall be set to 1.

If the PlayItem is not the first one in the PlayList, the meanings of the connection_condition are defined in the following.

connection_condition=1 –

This connection is not meant for seamless connection.

The constraints on this condition are:

- If the *PL_CPI_type* defined in the PlayList() indicates the *EP_map type of PlayList*, the *OUT_time* of the previous PlayItem shall point to a presentation-time value in the interval from the *presentation_start_time[atc_id][stc_id]* until the *presentation_end_time[atc_id][stc_id]* (see 4.4.3.2) for the STC-sequence that is referred to by the *ref_to_STC_id* of the previous PlayItem, and the *IN_time* of the current PlayItem shall point to a presentation-time value in the interval from the *presentation_start_time[atc_id][stc_id]* until the *presentation_end_time[atc_id][stc_id]* (see 4.4.3.2) for the STC-sequence that is referred to by the *ref_to_STC_id* of the current PlayItem.
- If the *PL_CPI_type* defined in the PlayList() indicates the *TU_map type of PlayList*, the *OUT_time* of the previous PlayItem shall point to a time value on *TU_time_base* (see 4.4.5.9) for an ATC_sequence in the Clip that is referred to by the Clip_information_file_name of the previous PlayItem, and the *IN_time* of the current PlayItem shall point to a time value on *TU_time_base* (see 4.4.5.9) for an ATC_sequence in the Clip that is referred to by the Clip_information_file_name of the current PlayItem.

It occurs if one of the following cases is true.

- This connection may be created simply by setting the *OUT_time* of the previous PlayItem and the *IN_time* of the current PlayItem, or
- The *CPI_type* in the PlayList is *TU_map type.*

This connection is not meant for seamless connection, but if following two conditions are satisfied, seamless connection between the PlayItems may be realized.

(1) The OUT_time of the previous PlayItem and the IN_time of the current PlayItem are equal, and

(2) The *ref_to_STC_id* of the previous PlayItem and the *ref_to_STC_id* of the current PlayItem refer to the same STC-sequence.

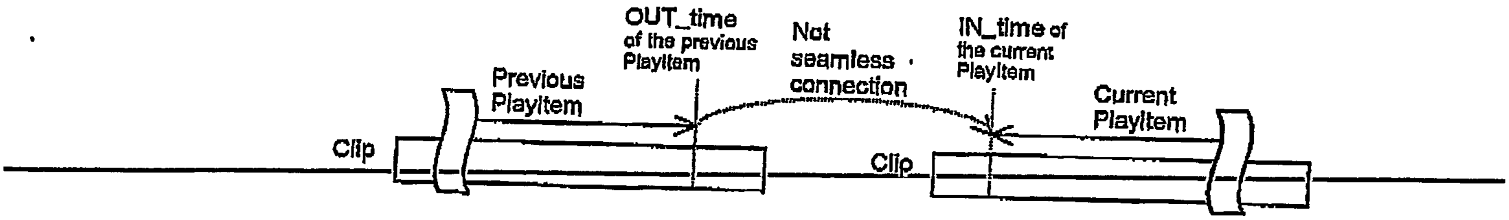


Figure 4-6 -- Connection of PlayItems with connection_condition set to 1

connection_condition=2 --

This connection is not meant for seamless connection.

The constraints on this condition are:

- This condition is permitted only if the *PL_CPI_type* defined in the PlayList() indicates the *EP_map type of PlayList.*

Consider the connection of two STC-sequences, those are STC-sequence1 and STC-sequence2. The STC-sequence1 is referred to by the *ref_to_STC_id* of the previous PlayItem. The STC-sequence2 is referred to by the *ref_to_STC_id* of the current PlayItem.

- The STC-sequence1 and the STC-sequence2 shall belong to the same ATC-sequence. The two STC-sequences shall be consecutive on the ATC-sequence.
- The *OUT_time* of the previous PlayItem shall be equal to the *presentation_end_time* (see 4.4.3.2) for the STC-sequence1. The *IN_time* of the current PlayItem shall be equal to the *presentation_start_time* (see 4.4.3.2) for the STC-sequence2.

It occurs in the following case.

- When an STC discontinuity occurs in a continuous recording of AV stream, the two PlayItems are divided for the STC discontinuity, but the ATC-sequence is continuous through the STC discontinuity.

An example of playing the PlayItems connected with this condition.

- Source packets used by the two PlayItems may be played continuously through the STC discontinuity, according to the continuous time base from the arrival time stamps of source packets. The player can reproduce a transport stream with the same arrival time base (through the STC discontinuity) as the arrival time base of the original transport stream at the recording.

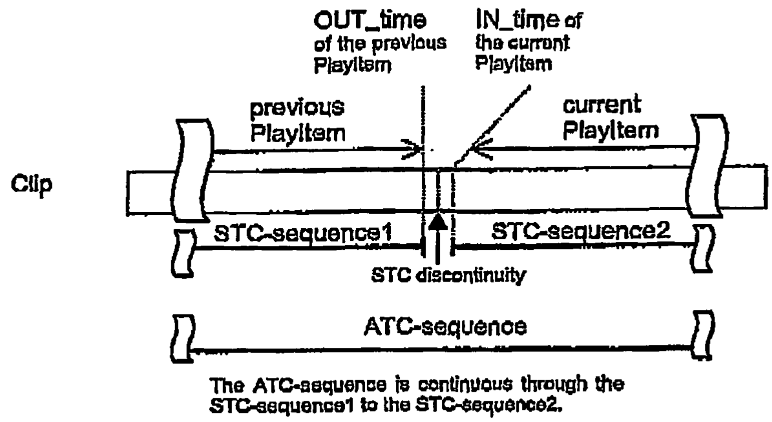


Figure 4-7 -- Connection of PlayItems with connection_condition set to 2

connection_condition=3 –

This connection is meant for seamless connection.

The constraints on this condition are:

- This condition is permitted only if the *PL_CPI_type* defined in the PlayList() indicates the *EP_map type of PlayList.*
- This condition is permitted only for the Virtual-PlayList.
- The previous PlayItem and the current PlayItem are connected with the Bridge-Clip, and there is a clean break at the connection point (See 6.3). The AV streams at the connection point shall comply with the constraints defined in chapter 6.
- The OUT_time of the previous PlayItem shall point to a presentation end time of the last video presentation unit (in presentation order) in the first ATC-sequence of the Bridge-Clip AV stream file specified by the *BridgeSequenceInfo* of the current PlayItem.
- The IN_time of the current PlayItem shall point to a presentation start time of the first video presentation unit (in presentation order) in the second ATC-sequence of the Bridge-Clip AV stream file specified by the *BridgeSequenceInfo* of the current PlayItem.

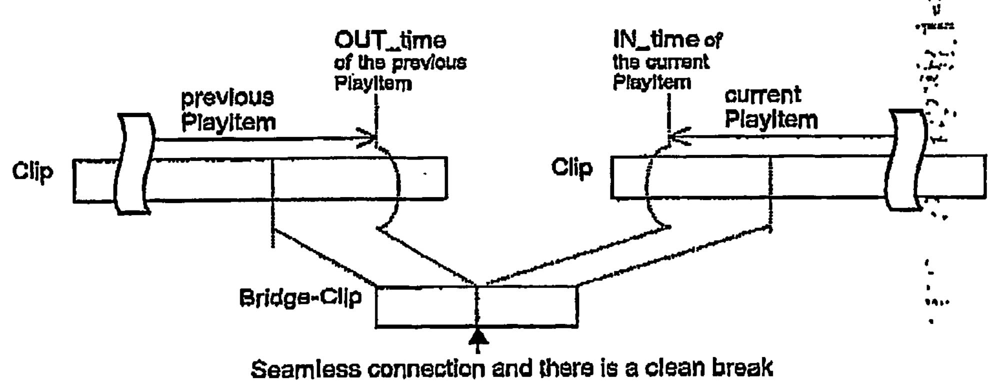


Figure 4-8-- Connection of PlayItems with connection_condition set to 3

connection_condition=4 –
This connection is meant for seamless connection.

The constraints on this condition are:

- This condition is permitted only if the *PL_CPI_type* defined in the PlayList() indicates the *EP_map type of PlayList*.

Consider the connection of two ATC-sequences, those are ATC-sequence1 and ATC-sequence2. The ATC-sequence1 is an ATC-sequence of the Clip that is referred to by the Clip_information_file_name of the previous PlayItem. The ATC-sequence2 is an ATC-sequence of the Clip that is referred to by the Clip_information_file_name of the current PlayItem.

- The previous PlayItem and the current PlayItem are connected without the Bridge-Clip, and there is a clean break at the connection point (See 6.4). The AV streams at the connection point shall comply with the constraints defined in chapter 6.
- The OUT_time of the previous PlayItem shall point to a presentation end time of the last video presentation unit (in presentation order) in the ATC-sequence1.
- The IN_time of the current PlayItem shall point to a presentation start time of the first video presentation unit (in presentation order) in the ATC-sequence2.
- The ATC-sequence1 and the ATC-sequence2 may be contained in the same Clip.

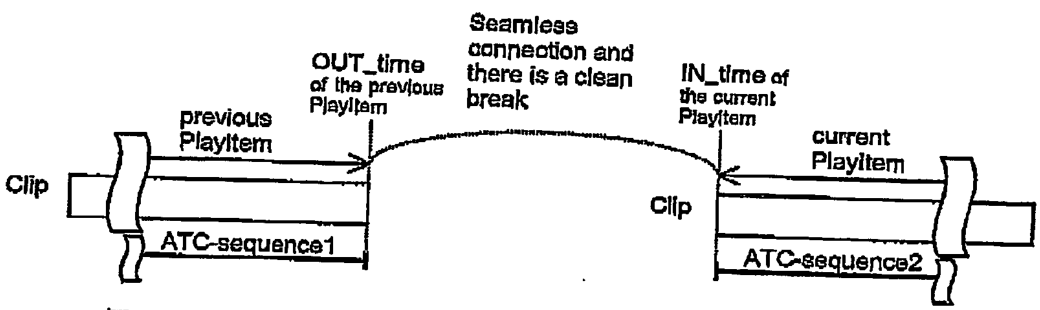


Figure 4-9 – Connection of PlayItems with connection_condition set to 4

ref_to_STC_id: If CPI_type in CPI() of the Clip information file referred to by the Clip_information_file_name is EP_map type, this 8-bit field indicates the *stc_id* value for an STC-sequence that contains presentation units for the current PlayItem. The Clip that is referred to by the Clip_information_file_name of the current PlayItem shall have the STC-sequence. The *stc_id* value is defined in the *SequenceInfo()* of the Clip.
See also the semantics of *connection_condition*.

IN_time: This is the IN_time of the current PlayItem. This 32-bit field indicates the presentation start time of the current PlayItem. The semantics of IN_time is different for the CPI_type in CPI() of the Clip information file referred to by the Clip_information_file_name as described in Table 4-7.

OUT_time: This is the OUT_time of the current PlayItem. This 32-bit field indicates the presentation end time of the current PlayItem. The semantics of OUT_time is different for the CPI_type in CPI() of the Clip information file referred to by the Clip_information_file_name as described in Table 4-7.

Table 4-7 – IN_time and OUT_time

CPI_type	Semantics of IN_time and OUT_time
EP_map type	Constraints on the *IN_time* and *OUT_time* of the current PlayItem are: • Each *IN_time* and *OUT_time* shall point to a presentation time derived from the STC time base on the Clip used by the PlayItem. • The *IN_time* and *OUT_time* are measured in units of a 45kHz clock. For example, each IN_time and OUT_time may be an upper 32-bit value of the 33-bit PTS (that has 90kHz accuracy) for a presentation unit. • The time interval from the *IN_time* to the *OUT_time* on the STC time base shall not have system time base discontinuities. • The *OUT_time* shall point to more future presentation time than the *IN_time*. If the STC in the STC-sequence is wrap-around between the *IN_time* and the *OUT_time*, the *IN_time* is greater than the *OUT_time*. • Additional constraints are specified in the semantics of *connection_condition* and section 4.3.4.3.1.
TU_map type	Constraints on the *IN_time* and *OUT_time* of the current PlayItem are: • The *IN_time* and *OUT*_time of the PlayItem shall point to times on the same *TU_time_base* for an ATC-sequence in the Clip that is referred to by the *Clip_information_file_name* of the PlayItem. The *TU_time_base* is defined in 4.4.5.11 (see semantics of SPN_time_unit_start*[atc_id][i]*). • The *IN_time* and *OUT_time* are measured in units of a 45kHz clock. • The *OUT_time* shall be greater than the *IN_time*. • Additional constraints are specified in the semantics of *connection_condition* and section 4.3.4.3.1.2.

4.3.4.3.1 Further constraints on IN_time and OUT_time of the PlayItem

Further constraints on IN_time and OUT_time of the PlayItem are described in the following.

4.3.4.3.1.1 PlayItem of the EP_map type of Real PlayList

Presentation times that are not used by any Real PlayLists of the EP_map type shall not exist in the interval from presentation_start_time[atc_id][stc_id] until presentation_end_time[atc_id][stc_id] of an STC-sequence of a Clip (not a Bridge-Clip) of the EP_map type. This rule shall be applied to all the STC-sequences of Clips of the EP-map type in the BDAV directory. IN_time and OUT_time of the EP_map type of Real PlayList shall be set to values to guarantee this restriction.

Suppose a Real PlayList of the EP_map type is created at the first recording, and the PlayList uses a Clip of the EP-map type, and the Clip has one STC-sequence with presentation_start_time[0][0] and presentation_end_time[0][0]. In this case, for example, IN_time and OUT_time of the PlayList are equal to the presentation_start_time[0][0] and the presentation_end_time[0][0] respectively.

4.3.4.3.1.2 PlayItem of the TU_map type of Real PlayList

Times that are not used by any Real PlayLists of the TU_map type shall not exist in the interval from arrival_start_time[atc_id] until arrival_end_time[atc_id] of the ATC-sequence of a Clip of the TU-map type. This rule shall be applied to all the ATC-sequences of Clips of the TU-map type in the BDAV directory. IN_time and OUT_time of the TU_map type of Real PlayList shall be set to values to guarantee this restriction.

4.3.4.3.1.3 Other restrictions for PlayItems of the EP_map type of PlayList

- When the current PlayItem has the connection_condition set to 3, the IN_time of the previous PlayItem shall not point to a presentation time in the Bridge-Clip indicated by the BridgeSequenceInfo of the current PlayItem.

- When the current PlayItem has the connection_condition set to 3, the OUT_time of the current PlayItem shall not point to a presentation time in the Bridge-Clip indicated by the BridgeSequenceInfo of the current PlayItem.

- The first PlayItem in the PlayList shall not have a BridgeSequenceInfo. So the IN_time of the first PlayItem never point to a presentation time in a Bridge-Clip.

- The OUT_time of the last PlayItem in the PlayList never point to a presentation time in a Bridge-Clip.

4.3.4.3.2 Length of the playing interval of PlayItem

The PlayItem specifies a time based playing interval defined by the IN_time and the OUT_time of the PlayItem. This section defines the method for calculating the length of the playing interval of PlayItem. The *PlayList_duration* field defined in the UIAppInfoPlayList of the PlayList file indicates the sum of playing intervals for all PlayItems in the PlayList.

If the *PL_CPI_type* defined in the PlayList() indicates the *EP_map type of PlayList*, the length of the playing interval of PlayItem is calculated by the following equations:

$L = OUT\_time - IN\_time$; if the OUT_time is greater than the IN_time.

$L = 0x100000000 - IN\_time + OUT\_time$; if the OUT_time is less than the IN_time.

If the *PL_CPI_type* defined in the PlayList() indicates the *TU_map type of PlayList*, the length of the playing interval of PlayItem is calculated by the following equations:

$L = OUT\_time - IN\_time$

Here the L is the length of the playing interval of PlayItem. The 32-bit unsigned integer measured in units of (1/45000) seconds.

4.3.5 BridgeSequenceInfo

This section defines the syntax and semantics of **BridgeSequenceInfo()** in the PlayItem().

4.3.5.1 BridgeSequenceInfo - General

- The *BridgeSequenceInfo()* is an attribute for the **current PlayItem** (see 4.3.4.1.1).
- The *BridgeSequenceInfo()* contains **Bridge_Clip_Information_file_name** to specify a Bridge-Clip AV stream file and the associated Clip Information file.
- The ***SPN_exit_from_preceding_Clip*** is a source packet number of a source packet in the Clip1 shown in the Figure 4-10. And the end of the source packet is the point where the player exits from the Clip1 to the start of the Bridge-Clip AV stream file. This is defined in the *ClipInfo() of the Bridge-Clip.*
- The ***SPN_enter_to_following_Clip*** is a source packet number of a source packet in the Clip2 shown in the Figure 4-10. And the start of the source packet is the point where the player enters to the Clip2 from the end of the Bridge-Clip AV stream file. This is defined in the *ClipInfo() of the Bridge-Clip.*
- The Bridge-Clip AV stream file contains two ATC-sequences.
- Clip1 and Clip2 can be the same Clip.

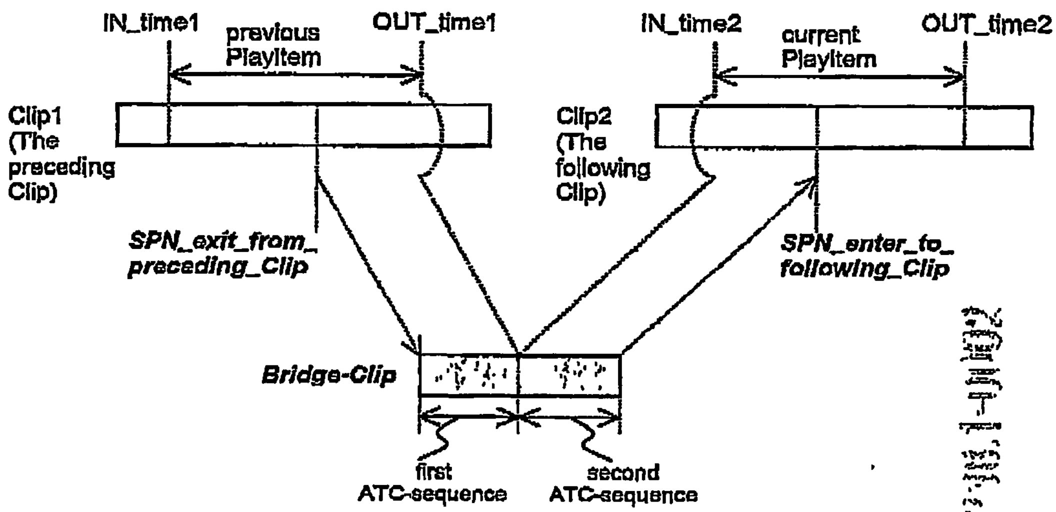


Figure 4-10 – An example of BridgeSequenceInfo

4.3.5.2 BridgeSequenceInfo - Syntax

Syntax	No. of bits	Mnemonic
BridgeSequenceInfo() {		
Bridge_Clip_Information_file_name	8*5	bslbf
Clip_codec_identifier	8*4	bslbf
reserved_for_future_use	8	bslbf
}		

4.3.5.3 BridgeSequenceInfo() - Semantics

Bridge_Clip_Information_file_name: This field specifies the name of a Clip information file for the Bridge-Clip used by the BridgeSequenceInfo. This field shall contain the 5-digit number "zzzzz" of the name of the Clip except the extension. It shall be coded according to ISO 646. The *Clip_stream_type* field in the *ClipInfo* of the Clip information file shall indicate "a Bridge-Clip AV stream of the BDAV MPEG-2 transport stream".

Clip_codec_identifier: This field shall have the value "M2TS" coded according to ISO 646 in the PlayList file using BDAV MPEG-2 transport streams.

The Bridge-Clip used by the BridgeSequenceInfo shall have an EP_map in the CPI().

4.4 Clip information file for the BDAV MPEG-2 transport stream

This section describes the syntax and semantics of the *zzzzz.clpi* file.

4.4.1 zzzzz.clpi (for the BDAV MPEG-2 transport stream)

4.4.1.1 zzzzz.clpi (for the BDAV MPEG-2 transport stream) - General

The zzzzz.clpi for the BDAV MPEG-2 transport stream is composed of six objects, and those are *ClipInfo(), SequenceInfo(), ProgramInfo(), CPI(), ClipMark()* and *MakersPrivateData()*.

The same 5-digit number "zzzzz" shall be used for both one AV stream file (a Clip AV stream file or a Bridge-Clip AV stream file) and the associated Clip information file.

4.4.1.2 zzzzz.clpi (for the BDAV MPEG-2 transport stream) - Syntax

Syntax	No. of bits	Mnemonic
zzzzz.clpi {		
type_indicator	8*4	bslbf
version_number	8*4	bslbf
SequenceInfo_start_address	32	uimsbf
ProgramInfo_start_address	32	uimsbf
CPI_start_address	32	uimsbf
ClipMark_start_address	32	uimsbf
MakersPrivateData_start_address	32	uimsbf
reserved_for_future_use	96	bslbf
ClipInfo()		
for(i=0; i<N1; i++){		
padding_word	16	bslbf
}		
SequenceInfo()		
for(i=0; i<N2; i++){		
padding_word	16	bslbf
}		
ProgramInfo()		
for(i=0; i<N3; i++){		
padding_word	16	bslbf
}		
CPI()		
for(i=0; i<N4; i++){		
padding_word	16	bslbf
}		
ClipMark()		
for(i=0; i<N5; i++){		
padding_word	16	bslbf
}		
MakersPrivateData()		
for(i=0; i<N6; i++){		
padding_word	16	bslbf
}		
}		

4.4.1.3 zzzzz.clpi (for the BDAV MPEG-2 transport stream) - Semantics

type_indicator: This field shall have the value "M2TS" coded according to ISO 646.

version_number: A four-character string that indicates version number of the Clip Information file. The version_number shall have the value "0100" coded according to ISO 646.

SequenceInfo_start_address: It indicates the start address of the *SequenceInfo()* in relative byte number from the first byte of the Clip Information file. The relative byte number starts from zero.

ProgramInfo_start_address: It indicates the start address of the *ProgramInfo()* in relative byte number from the first byte of the Clip Information file. The relative byte number starts from zero.

CPI_start_address: It indicates the start address of the *CPI()* in relative byte number from the first byte of the Clip Information file. The relative byte number starts from zero.

ClipMark_start_address: It indicates the start address of the *ClipMark()* in relative byte number from the first byte of the Clip Information file. The relative byte number starts from zero.

MakersPrivateData_start_address: It indicates the start address of the *MakersPrivateData()* in relative byte number from the first byte of the Clip Information file. The relative byte number starts from zero. If this field is set to zero, there is no data for the *MakersPrivateData()*. This rule is applied only for the *MakersPrivateData_start_address*.

padding_word: Padding words shall be inserted according to the syntax of zzzzz.clpi. N1,N2,N3,N4,N5 and N6 shall be zero or arbitrary positive integers. Each padding_word may have any value.

4.4.2 ClipInfo

This section defines the syntax and semantics of **ClipInfo()** in the Clip Information file.

4.4.2.1 ClipInfo - General

The *ClipInfo()* stores the attributes of the associated AV stream file (the Clip AV stream or the Bridge-Clip AV stream).

4.4.2.2 ClipInfo - Syntax

Syntax	No. of bits	Mnemonic
ClipInfo() {		
length	32	uimsbf
reserved_for_future_use	16	bslbf
Clip_stream_type	8	uimsbf
reserved_for_word_align	4	bslbf
encode_condition	2	bslbf
transcode_mode_flag	1	bslbf
controlled_time_flag	1	bslbf
TS_average_rate	32	uimsbf
TS_recording_rate	32	uimsbf
num_of_source_packets	32	uimsbf
BD_system_use	1024	bslbf
TS_type_info_block()		
if (Clip_stream_type == 2) {		
preceding_Clip_Information_file_name	8*5	bslbf
Clip_codec_identifier	8*4	bslbf
reserved_for_future_use	8	bslbf
SPN_exit_from_preceding_Clip	32	uimsbf
following_Clip_Information_file_name	8*5	bslbf
Clip_codec_identifier	8*4	bslbf
reserved_for_future_use	8	bslbf
SPN_enter_to_following_Clip	32	uimsbf
}		
}		

4.4.2.3 ClipInfo - Semantics

length: This 32-bit field indicates the number of bytes of the *ClipInfo()* immediately following this *length* field and up to the end of the *ClipInfo()*.

Clip_stream_type: This 8-bit field indicates a type of the AV stream associated with the Clip Information file.

Table 4-11 – Clip_stream_type

Clip_stream_type	meaning
0	reserved for future use
1	A Clip AV stream of the BDAV MPEG-2 transport stream (defined in 5.3).
2	A Bridge-Clip AV stream of the BDAV MPEG-2 transport stream (defined in 5.4).
3 - 255	reserved for future use

encode_condition:
If the Format_identifier in the TS_type_info_block() is set to 0x53 45 53 46 (ASCII code of "SESF"), the encode_condition indicates an encoding condition of the SESF transport stream for the Clip as defined in Table 4-12.
If the Format_identifier in the TS_type_info_block() is not set to 0x53 45 53 46 (ASCII code of "SESF"), the encode_condition has no meaning and shall be set to 00_b.

Table 4-12 – encode_condition (Valid only for SESF transport streams)

encode_condition	meaning
00_b	The SESF transport stream for the Clip does not comply with constraints specified in 8.9.
01_b	The SESF transport stream for the Clip shall comply with constraints on encode_condition=01_b, specified in 8.9.
10_b	reserved
11_b	The SESF transport stream for the Clip shall comply with constraints on encode_condition=11_b, specified in 8.9.

transcode_mode_flag: This flag indicates the recording way of MPEG-2 transport streams received from a digital broadcaster. If the Format_identifier in the TS_type_info_block() is set to 0x53 45 53 46 (ASCII code of "SESF"), this flag has no meaning and shall be set to zero.

If the Clip_stream_type indicates a Clip AV stream:
If this flag is set to 1_b, the AV stream file associated with the Clip shall be recorded in the way of transcode mode described in chapter 9, 10 and 11.

If this flag is set to 0_b, the AV stream file associated with the Clip shall be recorded in the way of transparent recording mode.

If the Clip_stream_type indicates a Bridge-Clip AV stream:
This field has no meaning and shall be set to zero.

controlled_time_flag:
If the Clip_stream_type indicates a Clip AV stream:
If the controlled_time_flag is set to 1_b, the AV stream file is recorded in the way of 'controlled time' recording and the AV stream file shall meet the equation (4-1) at the first recording.

$$TS\_average\_rate*192/188*(t-\alpha) \le size\_clip(t) \le TS\_average\_rate*192/188*(t+\alpha) \quad \dots (4\text{-}1)$$

Here,

- *TS_average_rate* – is the average bit-rate of the transport stream in AV stream file, expressed in units of bytes/second.
- *t* – is a time on the arrival time axis for the transport stream. The arrival time of the first transport packet in the AV stream file projects on to the origin where the *t* equals to zero in the arrival time axis. The *t* is measured in seconds.
- *size_clip(t)* – is the size of the AV stream file at time t, measured in bytes. For example, if there are 10 source packets from time 0 to t, the size_clip(t) is 10*192 bytes.
- α – is 300 seconds

If the *controlled_time_flag* is set to 0_b, the AV stream file is not recorded in the way of 'controlled time' recording, e.g., transparent recording of the input transport steam.

If the Clip_stream_type indicates a Bridge-Clip AV stream:
This field has no meaning and shall be set to zero.

TS_average_rate:
If the Clip_stream_type indicates a Clip AV stream:
If the *controlled_time_flag* is set to 1_b, this 32-bit field indicates the value of TS_average_rate using in the equation (4-1). If the *controlled_time_flag* is set to 0_b, this field has no meaning and shall be set to zero.

If the Clip_stream_type indicates a Bridge-Clip AV stream:
This field has no meaning and shall be set to zero.

TS_recording_rate:
If the Clip_stream_type indicates a Clip AV stream:
This 32-bit field indicates a value of the R_{MAX} defined by the equation (5-2) of 5.2.2. This value indicates the maximum data rate that the transport stream of AV stream file requires at the first recording. The TS_recording_rate is measured in units of bytes/second.

- In the case of SESF recording, the TS_recording_rate should be set the maximum rate of the transport stream.
- In the case of transparent recording through the digital interface, the TS_recording_rate may be set the allocated bandwidth for the input transport stream on the digital interface.
- In case of transmitting the transport stream over the digital interface, the TS_recording_rate may be used for allocating bandwidth on the digital interface.

If the *Clip_stream_type* indicates a Bridge-Clip AV stream:
This field has no meaning and shall be set to zero.
The *TS_recording_rate* for the first ATC-sequence in the Bridge-Clip shall be the same as the *TS_recording_rate* of the Clip connected ahead with the Bridge-Clip has.
The *TS_recording_rate* for the second ATC-sequence in the Bridge-Clip shall be the same as the *TS_recording_rate* of the Clip connected behind with the Bridge-Clip has.

num_of_source_packets: This field shall indicate the number of source packets stored in the AV stream file associated with the Clip Information file.

BD_system_use: This field contains the content protection information for the AV stream file associated with the Clip Information file.

For the application of part-3 using the system defined in Blu-ray Disc Content Protection System Specifications[8], additional constraints are specified in H.3 of Annex H.

preceding_Clip_Information_file_name: If the *Clip_stream_type* indicates the Clip is a Bridge-Clip AV stream file, then the *preceding_Clip_Information_file_name* specifies the name of a Clip Information file associated with a Clip AV stream file that is connected ahead with the Bridge-Clip AV stream file. This field shall contain the 5-digit number "zzzzz" of the name of the Clip except the extension. The name shall be coded according to ISO 646.
(See 4.3.5. The Clip indicated by this field is the *Clip-1* shown in Figure 4-10.)

SPN_exit_from_preceding_Clip: This 32-bit field indicates a source packet number of a source packet in a Clip specified by the *preceding_Clip_Information_file_name*. And the end of the source packet is the point where the player exits from the Clip to the start of the Bridge-Clip AV stream file. This means that the source packet pointed to by the *SPN_exit_from_preceding_Clip* is connected with the first source packet of the Bridge-Clip AV stream file. See 4.3.5.

following_Clip_Information_file_name: If the *Clip_stream_type* indicates the Clip is a Bridge-Clip AV stream file, then the *following_Clip_Information_file_name* specifies the name of a Clip Information file associated with a Clip AV stream file that is connected behind with the Bridge-Clip AV stream file. This

field shall contain the 5-digit number "zzzzz" of the name of the Clip except the extension. The name shall be coded according to ISO 646.
(See 4.3.5. The Clip indicated by this field is the *Clip-2* shown in Figure 4-10.)

SPN_enter_to_following_Clip: This 32-bit field indicates a source packet number of a source packet in a Clip specified by the *following_Clip_information_file_name*. And the start of the source packet is the point where the player enters to the Clip from the end of the Bridge-Clip AV stream file. This means that the last source packet of the Bridge-Clip AV stream file is connected with the source packet indicated by the *SPN_enter_to_following_Clip*. See 4.3.5.

Clip_codec_identifier: This field shall have the value "M2TS" coded according to ISO 646.

4.4.3 SequenceInfo

This section defines the syntax and semantics of **SequenceInfo()** in the Clip Information file.

4.4.3.1 SequenceInfo - General

The *SequenceInfo()* stores information to describe *ATC-sequences* and *STC-sequences* for the AV stream file.

4.4.3.1.1 ATC-sequence

- A time-line based on the arrival time of each source packet in the AV stream file is called an *ATC*. The sequence of source packets that includes no arrival time-base (ATC) discontinuity is called an *ATC-sequence*.
- When making a new recording of Clip AV stream file, the Clip shall contain no arrival time-base discontinuity, i.e. the Clip shall contain only one ATC-sequence.

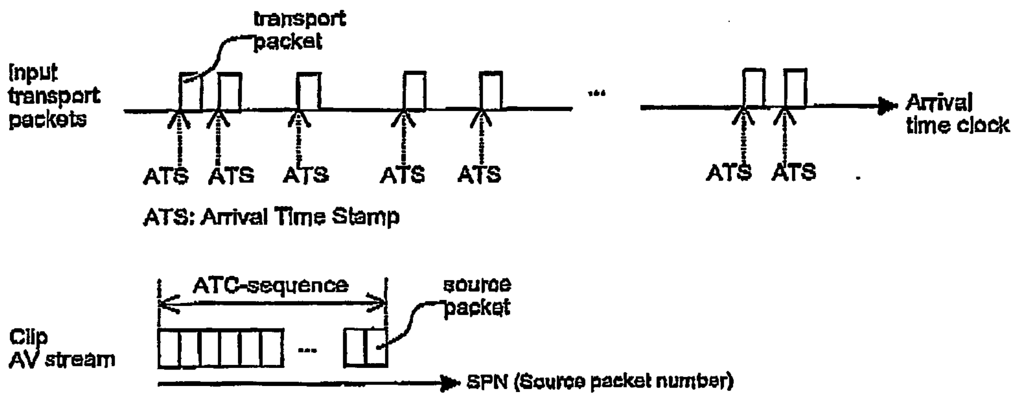


Figure 4-13 – An illustration of an ATC-sequence

- It is supposed that the arrival time base discontinuities in the Clip AV stream file may only occur in case the parts of the Clip AV stream are deleted by editing and the needed parts originated from the same Clip are combined into a new Clip AV stream file. See the example shown in Figure 2-8. In case of the editing shown in Figure 2-8, the number of ATC-sequences in a Clip is one before editing, and the number is changed to two after editing. Figure 4-14 illustrates the change of ATC-sequences in the Clip.
- The *SequenceInfo()* stores addresses where the arrival time-bases start. The *SPN_ATC_start* indicates the address.
- The *ATC-sequence* except the last one in the AV stream file starts from the source packet pointed to by the *SPN_ATC_start*, and ends at the source packet immediately before the source packet pointed to by the next *SPN_ATC_start*. The last ATC-sequence starts from the source packet pointed to by the last *SPN_ATC_start*, and ends at the last source packet. For example, the Clip shown in Figure 4-15 contains three ATC-sequences.
- The first source packet of the *ATC-sequence* shall be the first source packet of an Aligned unit.

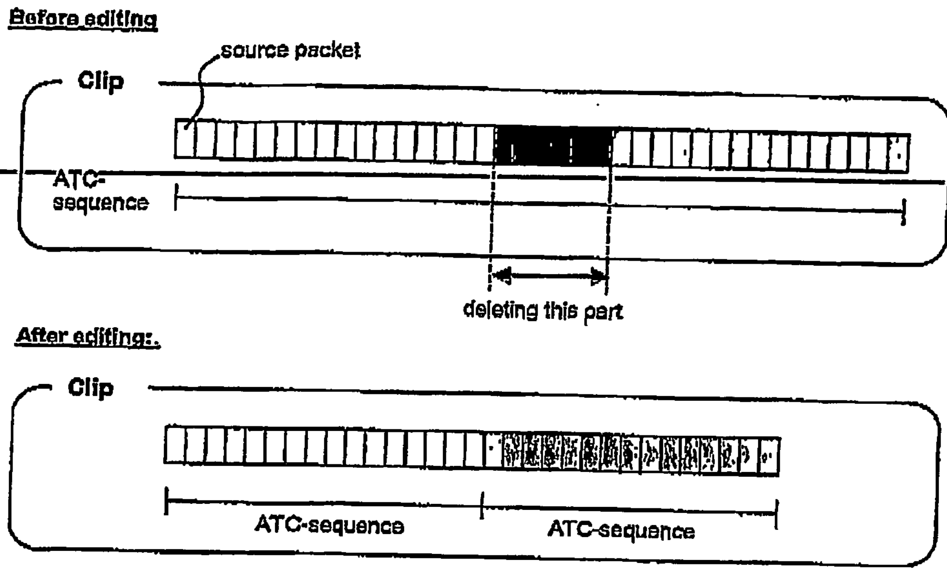


Figure 4-14 – An example of Clip that contains two ATC-sequences

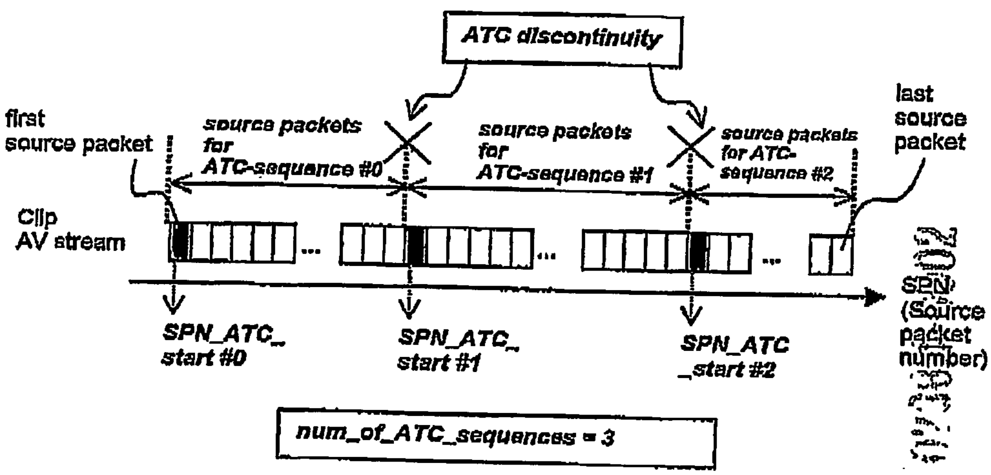


Figure 4-15 – An example of arrival time base discontinuities and ATC-sequences

4.4.3.1.2 STC-sequence

- The sequence of source packets that includes no STC discontinuity (system time-base discontinuity) is called an ***STC-sequence***. The 33-bit counter of STC may wrap-around in the STC-sequence (See the case-2 in Figure 4-16).

  Note that when the recorder records an AV stream (Clip AV stream file or Bridge-Clip AV stream) with an EP_map, the stream shall contain *a single program at one time*, namely *a single system time-base at one time.*

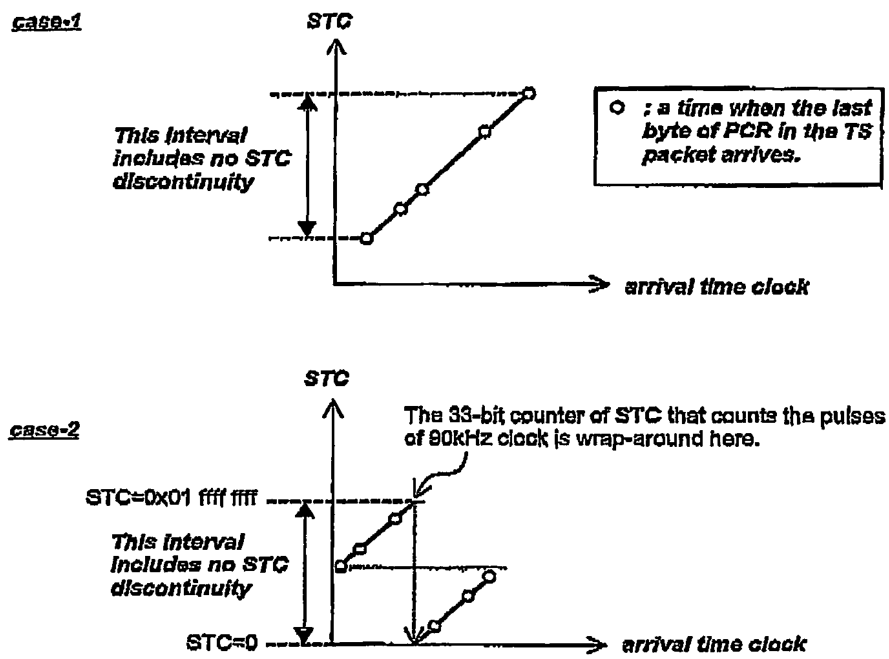


Figure 4-16 – An illustration of the continuous STC interval

- The *SequenceInfo()* stores addresses where the system time-bases start. The ***SPN_STC_start*** indicates the address.
- The *STC-sequence* except the last one in the AV stream file starts from the source packet pointed to by the *SPN_STC_start*, and ends at the source packet immediately before the source packet pointed to by the next *SPN_STC_start*. The last STC-sequence starts from the source packet pointed to by the last *SPN_STC_start*, and ends at the last source packet. For example, the Clip shown in Figure 4-17 contains three STC-sequences.
- No STC-sequence can overlap the ATC-sequence boundary.

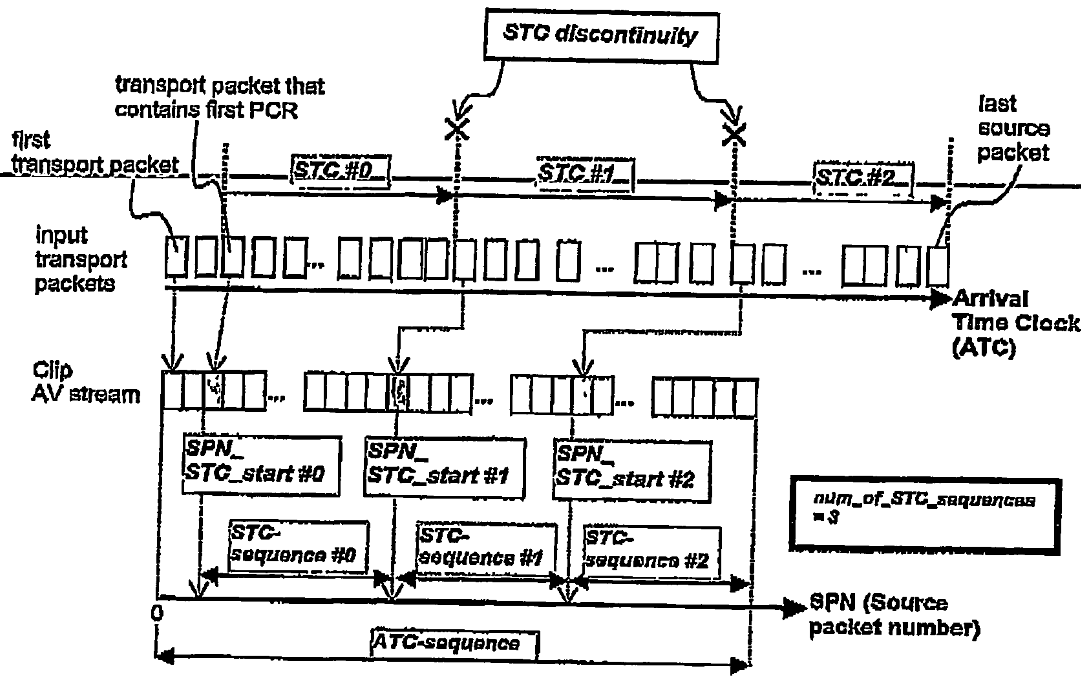


Figure 4-17 – An example of STC-sequence

4.4.3.2 SequenceInfo - Syntax

Syntax	No. of bits	Mnemonic
SequenceInfo() {		
length	32	uimsbf
reserved_for_word_align	8	bslbf
num_of_ATC_sequences	8	uimsbf
for (*atc_id=0; atc_id<num_of_ATC_sequences; atc_id++*) {		
SPN_ATC_start[*atc_id*]	32	uimsbf
num_of_STC_sequences[*atc_id*]	8	uimsbf
offset_STC_id[*atc_id*]	8	uimsbf
for (*stc_id = offset_STC_id[atc_id]; stc_id <(num_of_STC_sequences[atc_id]+offset_STC_id[atc_id]); stc_id++*) {		
PCR_PID[*atc_id*][*stc_id*]	16	uimsbf
SPN_STC_start[*atc_id*][*stc_id*]	32	uimsbf
presentation_start_time[*atc_id*][*stc_id*]	32	uimsbf
presentation_end_time[*atc_id*][*stc_id*]	32	uimsbf
}		
}		
}		

4.4.3.3 SequenceInfo - Semantics

length: This 32-bit field indicates the number of bytes of the *SequenceInfo()* immediately following this *length* field and up to the end of the *SequenceInfo()*.

num_of_ATC_sequences: This 8-bit field indicates the number of ATC-sequences in the AV stream file (Clip AV stream file or Bridge-Clip AV stream file). Further restrictions are described in the semantics of *SPN_ATC_start[atc_id]* below.

SPN_ATC_start[*atc_id*]: This 32-bit field indicates a source packet number of a source packet where the ATC-sequence pointed to by *atc_id* starts in the AV stream file.

- If the *Clip_stream_type* in the ClipInfo() indicates *'a Clip AV stream'*, the *num_of_ATC_sequences* shall be greater than or equal to one.
- If the *Clip_stream_type* in the ClipInfo() indicates *'a Bridge-Clip AV stream'*, the *num_of_ATC_sequences* shall be set to two. (See 6.5.5)
- If the *num_of_ATC_sequences* value is one, the *SPN_ATC_start[0]* shall be set to zero (it points to the first source packet in the AV stream file).
- If the *num_of_ATC_sequences* value is greater than one, the first *SPN_ATC_start[atc_id]* shall be set to zero (it points to the first source packet in the AV stream file), and the other *SPN_ATC_start[atc_id]* values shall point to the Aligned unit boundaries, i.e., each *SPN_ATC_start[atc_id]* shall point to the first source packet in an Aligned unit.
- The entries of *SPN_ATC_start* in the *SequenceInfo* shall be sorted in ascending order of the *SPN_ATC_start* values.
- The numerical formula below shall be satisfied for the entries of *SPN_ATC_start[atc_id] values.*

$$SPN\_ATC\_start[0] = 0$$

$$\forall (0 < atc\_id < num\_of\_ATC\_sequences)$$

$$SPN\_ATC\_start[atc\_id - 1] < SPN\_ATC\_start[atc\_id]$$

When making a new recording of Clip AV stream file, the Clip shall contain no arrival time-base discontinuity and the *num_of_ATC_sequences* shall be set to one.

It is supposed that the arrival time base discontinuities in the Clip AV stream file may only occur in case the parts of the Clip AV stream are deleted by editing and the needed parts originated from the same Clip are combined into a new Clip AV stream file. See the example described in Figure 2-8.

This specification does not intend to allow combining any Clips into one Clip. For example, combining Clips that have different values in the TS_type_info_block() is not permitted (e.g. Combining DESF type of Clip and ISDB type of Clip is not permitted), combining EP_map type of Clip and TU_map type of Clip is not permitted.

num_of_STC_sequences[*atc_id*]: This 8-bit field indicates the number of STC-sequences on the ATC-sequence pointed to by the *atc_id*.

- If the *CPI_type* in the CPI() indicates *EP_map type*, this field shall be set to a value that is greater than or equal to one.
- If the *CPI_type* in the CPI() indicates *TU_map type*, this field may be set to zero. If this field is set to zero, the *offset_STC_id[atc_id]* has no meaning and shall be set to zero.

offset_STC_id[*atc_id*]: This 8-bit field indicates the offset *stc_id* value for the first STC-sequence on the ATC-sequence pointed to by the *atc_id*.

- When making a new recording of Clip AV stream file, the *offset_STC_id[atc_id]* shall be set to zero (and the *num_of_ATC_sequences* shall be set to one).

- The *stc_id* value for the STC-sequence on the ATC-sequence pointed to by the *atc_id* is defined by the order described by the for-loops of *stc_id* in the syntax, starting with the *offset_STC_id[atc_id]*. The *stc_id* value is referred to by the *ref_to_STC_id* defined in the *PlayItem(), SubPlayItem()* and the *ClipMark()*.
- In the two consecutive ATC-sequences defined in the SequenceInfo(), the last *stc_id* value defined in the previous ATC-sequence and the first *stc_id* value defined in the following ATC-sequence may be the same value. If these two *stc_id* values are the same, PTSs of the same value shall not appear in the two STC-sequences indicated by the two *stc_id* values. Note that these two STC-sequences are different STC-sequences, but have the same *stc_id* value. (See the case of "After editing" in Figure 4-19.)
- The *stc_id* entries in the SequenceInfo() shall be sorted in ascending order of the *stc_id* values, the *offset_STC_id[atc_id]* entries shall be set to values to guarantee this restriction.
- The *offset_STC_id[atc_id]* plus the *num_of_STC_sequences[atc_id]* shall be less than or equal to 255.

Figure 4-18 illustrates an example of deleting the beginning part of a Clip. In this case, the *offset_STC_id[atc_id]* may take non-zero value.

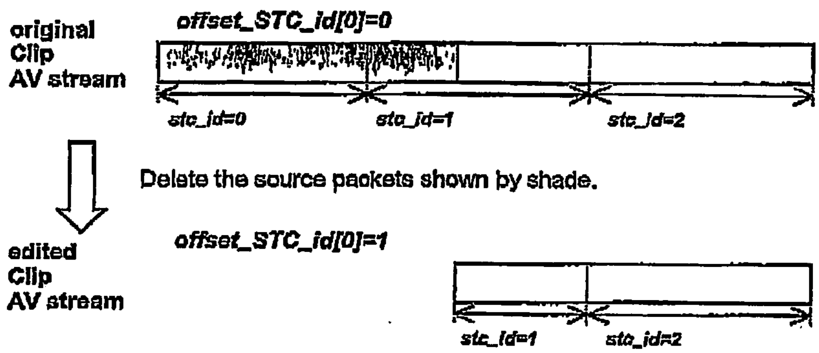


Figure 4-18 – An example case when the *offset_STC_id* takes non-zero value

Figure 4-19 illustrates an example of deleting the middle part of a Clip. Here, the STC-sequence with *stc_id=1* is used by the *PlayItem2* and the *PlayItem3*. When we delete the middle part of the STC_sequence with *stc_id=1* as shown in the figure, the result is shown in the figure of "After editing". The Clip after editing has two ATC-sequences, and the *offset_STC_id[0]* of the first ATC-sequence is set to zero and the *offset_STC_id[1]* of the second ATC-sequence is set to one. So the second STC-sequence (this is the last STC-sequence in the previous ATC-sequence) and the third STC-sequence (this is the first STC-sequence in the following ATC-sequence) have the same *stc_id* value (=1). Note that we do not need to change the values of *ref_to_STC_id* for the PlayItem3 and the PlayItem4. So when we delete parts of a Clip, we do not need to change the Virtual PlayLists that do not use the deleted parts of the Clip.

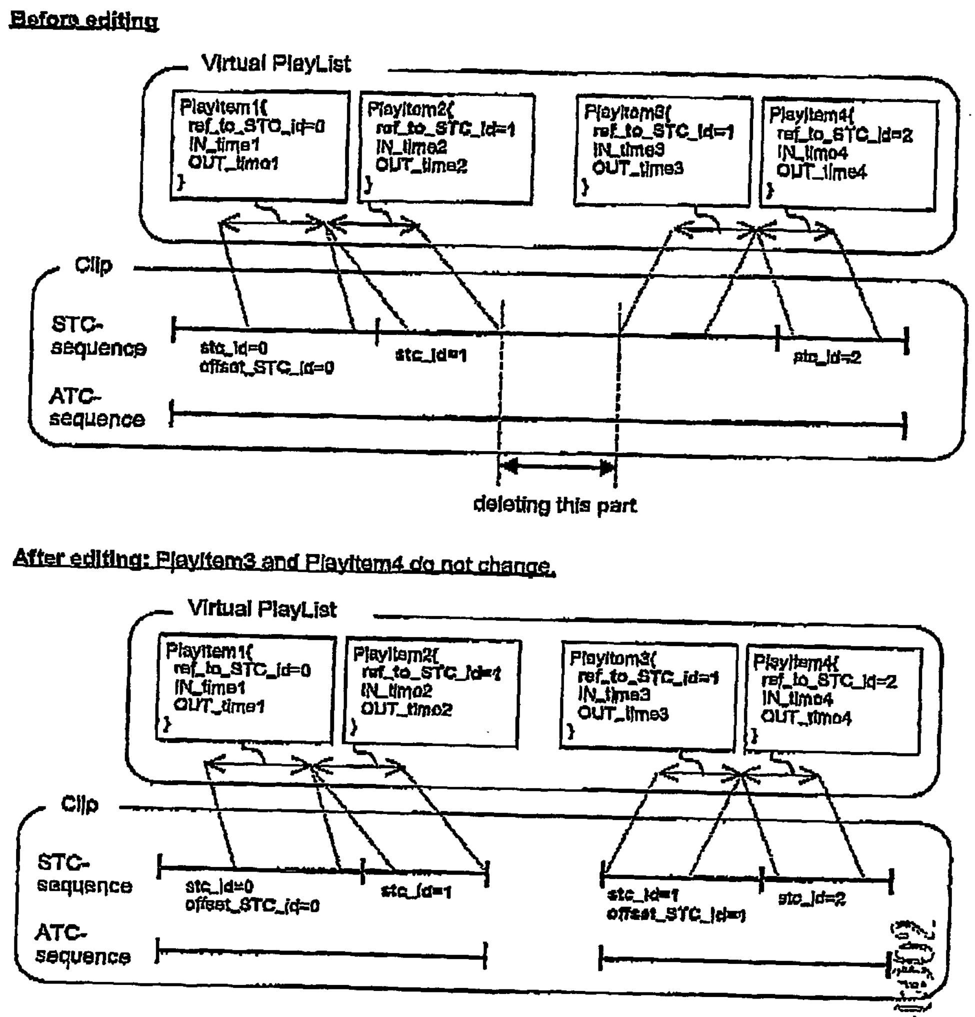


Figure 4-19 -- An example of deleting the middle part of Clip

PCR_PID*[atc_id][stc_id]*: This 16-bit field indicates the value of PID of the transport packets that shall contain the PCR fields valid for the *STC-sequence* pointed to by the *stc_id* on the ATC-sequence pointed to by the *atc_id*.

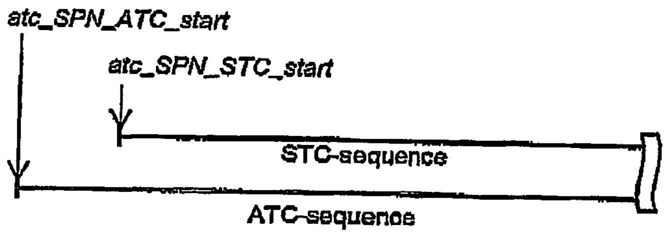
SPN_STC_start*[atc_id][stc_id]*: This 32-bit field indicates a source packet number of a source packet where the *STC-sequence* pointed to by the *stc_id* on the ATC-sequence pointed to by the *atc_id* starts.

- The entries of *SPN_STC_start[atc_id][stc_id]* in the *SequenceInfo()* shall be sorted in ascending order of the *SPN_STC_start[atc_id][stc_id]* values.

- The first *SPN_STC_start[atc_id][stc_id]* value in the ATC-sequence pointed to by *atc_id* shall be greater than or equal to the *SPN_ATC_start[atc_id]* value.

$$SPN\_ATC\_start[atc\_id] \leq SPN\_STC\_start[atc\_id][0]$$

The tolerance from the source packet pointed to by the *SPN_ATC_start[atc_id]* to the source packet pointed to by the first *SPN_STC_start[atc_id][stc_id]* on the ATC-sequence shall be less than or equal to *one second* in the arrival time base (See Figure 4-20).



atc_SPN_STC_start - atc_SPN_ATC_start <= 1 seconds

Here,
atc_SPN_ATC_start is the arrival time of the source packet pointed to by the *SPN_ATC_start[atc_id]*.
atc_SPN_STC_start is the arrival time of the source packet pointed to by the first *SPN_STC_start[atc_id][stc_id]* on the ATC-sequence.

Figure 4-20 – The *SPN_ATC_start[atc_id]* and the first *SPN_STC_start[atc_id][stc_id]*

- The *SPN_STC_start[atc_id][stc_id]* value should point to the source packet where system time-base discontinuities occur in the AV stream file.

ISO/IEC13818-1[4] describes the transmission of the MPEG-2 transport streams. Here the system time-base discontinuity point is defined to be the instant in time when the first byte of a transport packet containing a PCR of a new system time-base arrives at the input of the T-STD (See semantics of discontinuity_indicator).

New discontinuities can be introduced when using storage systems (editing of recorded transport streams or using special playback modes). That is why for Storage-Media-Interoperability, a more accurate method is specified for indicating the location of the discontinuity. For this purpose the DIT transport packet is applied (see EN 300 468[13]).

The *SPN_STC_start* should point to the location where the discontinuity occurs. This is either:

- A known location (discontinuity point in the Bridge-Clip, copying parts of Clips into a new Clip).
- Indicated with a DIT table. Then the *SPN_STC_start* points to the location of the DIT packet (the first one from a pair of two consecutive DIT packets inserted at the discontinuity point[13]).
- Derived by parsing PCR packets. Here the MPEG-2 system time-base discontinuity point is found.

There is a tolerance on the value of the *SPN_STC_start* because:

- There might be a delay in detecting the presence of a DIT packet.
- Several PCR packets of the new STC are needed before it is concluded that there is a STC discontinuity in the TS.

It is recommended to keep the tolerance small. This can be obtained by remembering the source packet number of the DIT or the PCR packet where the first discontinuity occurred.

ISO/IEC 13818-1[6] defines the PCR tolerance as the maximum inaccuracy allowed in received PCR. The PCR tolerance is ± 500 nano-seconds (See ISO/IEC 13818-1, section 2.4.2.2). When the input TS complies with ISO/IEC 13818-1 and the received PCR has an inaccuracy more than the PCR tolerance, the PCR packet should be a system time-base discontinuity point.

ISO/IEC 13818-9[7] also defines the PCR tolerance. In case of real time interface for low jitter applications, the PCR tolerance is ± 25 micro-seconds. When the input TS complies with the case and received PCR has inaccuracy more than the PCR tolerance, the PCR packet should be a system time-base discontinuity point.

presentation_start_time*[atc_id][stc_id]*: This 32-bit field indicates a presentation start time of the AV stream data for the *STC-sequence* pointed to by the *stc_id* on the ATC-sequence pointed to by the *atc_id*. This value is a presentation time measured in units of a 45kHz clock derived from the STC of the *STC-sequence.*

- If the *Clip_stream_type* in the *ClipInfo()* indicates *a Clip AV stream* and the *CPI_type* in the *CPI()* indicates *EP_map type* :

  the *presentation_start_time[atc_id][stc_id]* shall satisfy both (1) and (2) conditions as follows.

  (1) the *presentation_start_time[atc_id][stc_id]* shall point to a time within a time-segment that consists of a presentation time pointed to by the first *PTS_EP_fine* entry (in the *EP_map*) on the *STC-sequence* and the following 5 seconds after the *PTS_EP_fine* entry (a SPN derived from *SPN_EP_fine* associated with the first *PTS_EP_fine* on the *STC-sequence* shall be greater than or equal to *SPN_STC_start* of the STC-sequence.).

  and

  (2) the *presentation_start_time[atc_id][stc_id]* shall satisfy either of the following two conditions.

  If the first *PTS_EP_fine* entry on the STC-sequence points to a video access unit (the *EP_stream_type* for the *PTS_EP_fine* indicates 1 or 2), the *presentation_start_time [atc_id][stc_id]* shall point to a time within a time-segment that consists of a presentation start time of the first video presentation unit (in presentation order) *on the STC-sequence* and the following 5 seconds after the presentation start time.

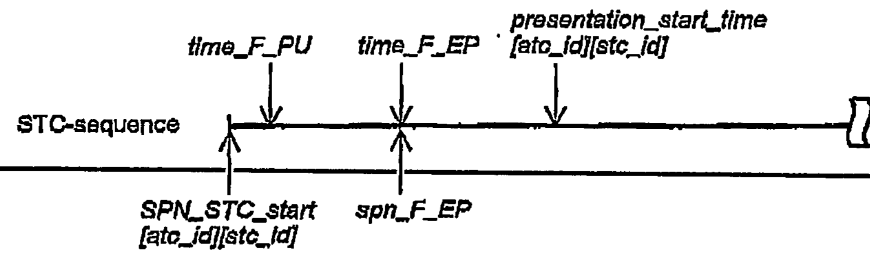
  or

  If the first *PTS_EP_fine* entry on the STC-sequence points to an audio access unit (the *EP_stream_type* for the *PTS_EP_fine* indicates 3), the *presentation_start_time [atc_id][stc_id]* shall point to a time within a time-segment that consists of a presentation start time of the first audio presentation unit (in presentation order) *on the STC-sequence* and the following 5 seconds after the presentation start time.

  (See an example shown in Figure 4-21.)

- If the Clip_stream_type in the ClipInfo() indicates a Bridge-Clip AV stream :
  the *presentation_start_time[atc_id][stc_id]* shall comply with the constraints described in 4.4.3.3.1.

- If the *Clip_stream_type* in the *ClipInfo()* indicates *a Clip AV stream* and the *CPI_type* in the *CPI()* indicates *TU_map type* :
  the *presentation_start_time[atc_id][stc_id]* is optional and may have any value (Note that the *num_of_STC_sequences[atc_id]* may be set to zero in this case).



presentation_start_time[atc_id][stc_id] - time_F_EP $\leq$ ***5 seconds***
&&
presentation_start_time[atc_id][stc_id] - time_F_PU $\leq$ ***5 seconds***
&&
SPN_STC_start[atc_id][stc_id] $\leq$ ***spn_F_EP***

Here,
time_F_EP is the presentation time pointed to by the first *PTS_EP_fine* entry on the STC-sequence.
time_F_PU is the presentation start time of the first video presentation unit (in presentation order) on the STC-sequence.
spn_F_EP is the SPN pointed to by *SPN_EP_fine* associated with the first *PTS_EP_fine* on the STC-sequence.

Note that, in this example, there is not a wrap-around in STC between the *time_F_PU* and the *presentation_start_time[atc_id][stc_id]*.

Figure 4-21 – An example of *presentation_start_time[atc_id][stc_id]*

presentation_end_time*[atc_id][stc_id]*: This 32-bit field indicates a presentation end time of the AV stream data for the *STC-sequence* pointed to by the *stc_id* on the ATC-sequence pointed to by the *atc_id*. This value is a presentation time measured in units of a 45kHz clock derived from the STC of the *STC-sequence*.

The *presentation_end_time[atc_id][stc_id]* shall point to more future presentation-time than the *presentation_start_time[atc_id][stc_id]* in the STC-sequence. If the STC is wrap-around in the STC-sequence, the *presentation_start_time[atc_id][stc_id]* value may be greater than the *presentation_end_time[atc_id][stc_id]*.

- If the *Clip_stream_type* in the *ClipInfo()* indicates *a Clip AV stream* and the *CPI_type* in the *CPI()* indicates *EP_map type* :

  the *presentation_end_time[atc_id][stc_id]* shall satisfy either of the following two conditions.

  If a *PTS_EP_fine* entry that points to the nearest past presentation time from the *presentation_end_time[atc_id][stc_id]* points to a video access unit (the *EP_stream_type* indicates 1 or 2) on the STC-sequence, the presentation_end_time*[atc_id][stc_id]* shall point to a time within a time-segment that consists of a presentation end time of the last video presentation unit (in presentation order) on the *STC-sequence* and the preceding 5 seconds before the presentation end time.

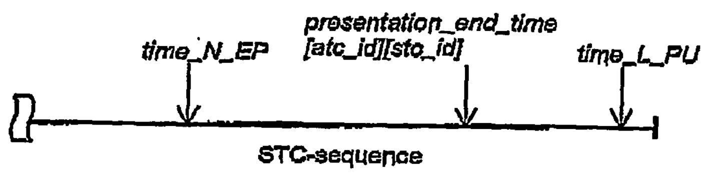
  or

  If a *PTS_EP_fine* entry that points to the nearest past presentation time from the *presentation_end_time[atc_id][stc_id]* points to an audio access unit (the *EP_stream_type* indicates 3) on the STC-sequence, the presentation_end_time *[atc_id][stc_id]* shall point to a time within a time-segment that consists of a presentation end time of the last audio presentation unit

(in presentation order) on the *STC-sequence* and the preceding 5 seconds before the presentation end time.
(See an example shown in Figure 4-22)

- If the Clip_stream_type in the ClipInfo() indicates a Bridge-Clip AV stream :
  the *presentation_end_time[atc_id][stc_id]* shall comply with the constraints described in 4.4.3.3.1.

- If the *Clip_stream_type* in the *ClipInfo()* indicates *a Clip AV stream* and the *CPI_type* in the *CPI()* indicates *TU_map type* :
  the *presentation_end_time[atc_id][stc_id]* is optional and may have any value (Note that the *num_of_STC_sequences[stc_id]* may be set to zero in this case).



time_L_PU - presentation_end_time[atc_id][stc_id] ≤ 5 seconds

Here,
time_N_EP is the presentation time pointed to by the *PTS_EP_fine* entry that points to the nearest past presentation time from the presentation_end_time[atc_id][stc_id] on the STC-sequence.
time_L_PU is the presentation end time of the last video presentation unit (in presentation order) on the STC-sequence.

Note that, in this example, there is not a wrap-around in STC between the *presentation_end_time[atc_id][stc_id]* and the *time_L_PU*.

Figure 4-22 – An example of *presentation_end_time[atc_id][stc_id]*

4.4.3.3.1 Additional constraints on SequenceInfo for Bridge-Clip AV stream file

If the Clip_stream_type in the ClipInfo() indicates a Bridge-Clip AV stream, this section defines additional constraints to the section 4.4.3.3 on the SequenceInfo for the Bridge-Clip AV stream file.

num_of_ATC_sequences :
The num_of_ATC_sequences shall be set to 2.

SPN_ATC_start*[atc_id]* :
The SPN_ATC_start[0] shall be set to zero.
The SPN_ATC_start[1] shall comply with the constraint described in 6.6.5 (2).

num_of_STC_sequences[atc_id] ;
Both the num_of_STC_sequences[0] and the num_of_STC_sequences[1] shall be set to one.

offset_STC_id*[atc_id]* :
Both the offset_STC_id[0] and the *offset_STC_id*[1] shall be set to zero.

SPN_STC_start*[atc_id][stc_id]*:
The SPN_STC_start[0][0] shall be set to zero.
The SPN_STC_start[1][0] shall be equal to the SPN_ATC_start[1].

presentation_start_time*[atc_id][stc_id]*:
The presentation_start_time[0][0] shall point to the most past PTS (in presentation order) of a video access unit in the first ATC-sequence of the Bridge-Clip.

The presentation_start_time[1][0] shall point to a presentation start time of the first complete video presentation unit (in presentation order) in the second ATC-sequence of the Bridge-Clip.

presentation_end_time*[atc_id][stc_id]*;
The presentation_end_time[0][0] shall point to a presentation end time of the last complete video presentation unit (in presentation order) in the first ATC-sequence of the Bridge-Clip.

The presentation_end_time[1][0] shall point to the most future PTS (in presentation order) of a video access unit in the second ATC-sequence of the Bridge-Clip.

5.2 BDAV MPEG-2 transport stream

5.2.1 Structure of BDAV MPEG-2 transport stream

The AV stream file shall have the structure of BDAV MPEG-2 transport stream.

The BDAV MPEG-2 transport stream has the following data structure described in Figure 5-1. The BDAV MPEG-2 transport stream is constructed from an integer number of Aligned units.

1) The size of an Aligned unit is 6144 bytes (2048*3 bytes).
2) The Aligned unit starts from the first byte of source packets.
3) The length of a source packet is 192 bytes. One source packet consists of a TP_extra_header and a transport packet. The length of TP_extra_header is 4 bytes and the length of transport packet is 188 bytes.
4) One Aligned unit consists of 32 source packets.
5) The last Aligned unit in the BDAV MPEG-2 transport stream also consists of 32 source packets. So, the BDAV MPEG-2 transport stream terminates at the end of an Aligned unit.
6) If the last Aligned unit is not completely filled with input transport stream to be recorded on the volume, the remaining bytes shall be filled with source packets with Null packet (transport packet with PID=0x1FFF).

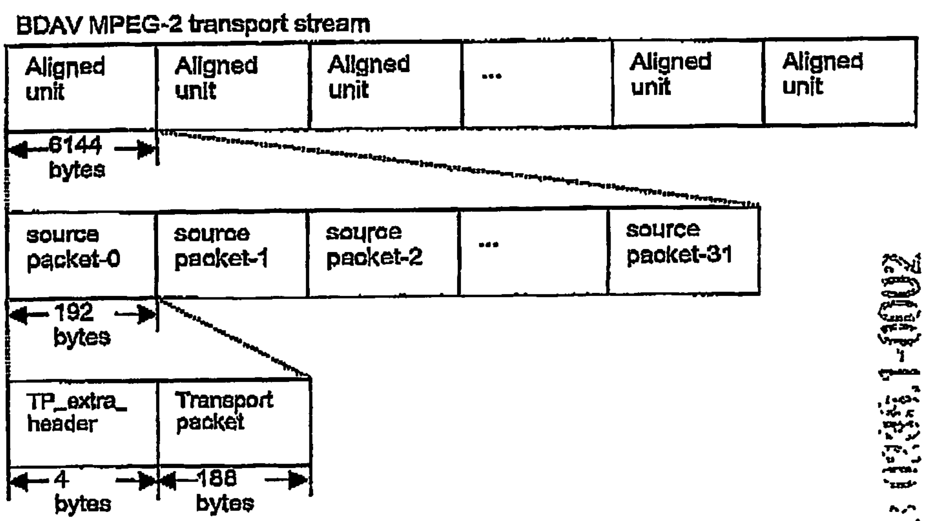


Figure 5-1 – Structure of BDAV MPEG-2 transport stream

CLAIMS:

1. Disc-like record carrier carrying a video signal which is reproducible via processing Playlist, indicating in which order parts of the video signal have to be reproduced, the PlayLists comprising PlayItems, each PlayItem referring to a part of the video signal, the part of the video signal comprising sequence blocks each block having a Source packet number

characterized in that if the PlayItem represents a bridge-clip the play-item further comprises a parameter indicative for the Source Packet numbers that have to be processed from the part of the video signal referred to be the current PlayItem, a reference to a Bridge Clip AV stream file, and a parameter indicating which Source packet numbers of the part of the video signal corresponding to the following PlayItem have to be processed.

2. Method for generating a disc like record carrier according to claim 1.

3. Apparatus for reproducing a video signal recorded on a record carrier according to claim 1

4. Method for reproducing a video signal recorded on a record carrier according to claim 1.

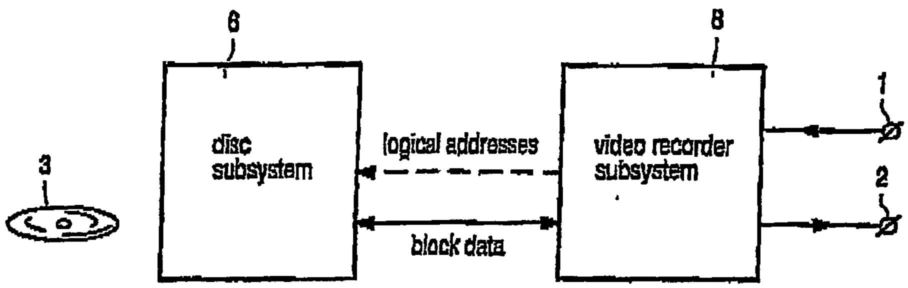


FIG. 1

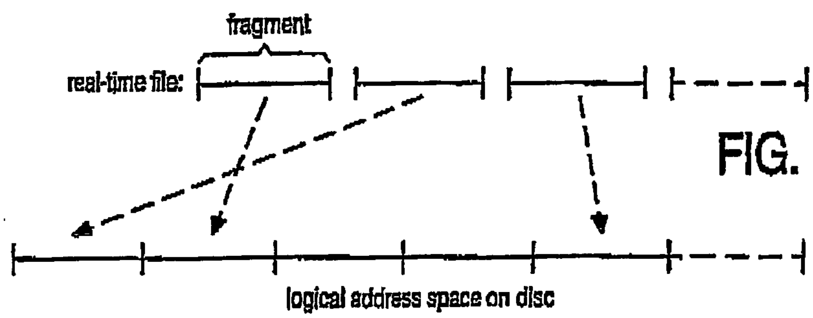


FIG. 2a

FIG. 2b

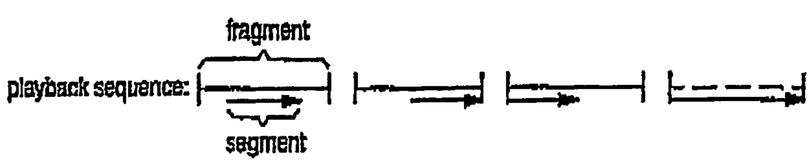


FIG. 3

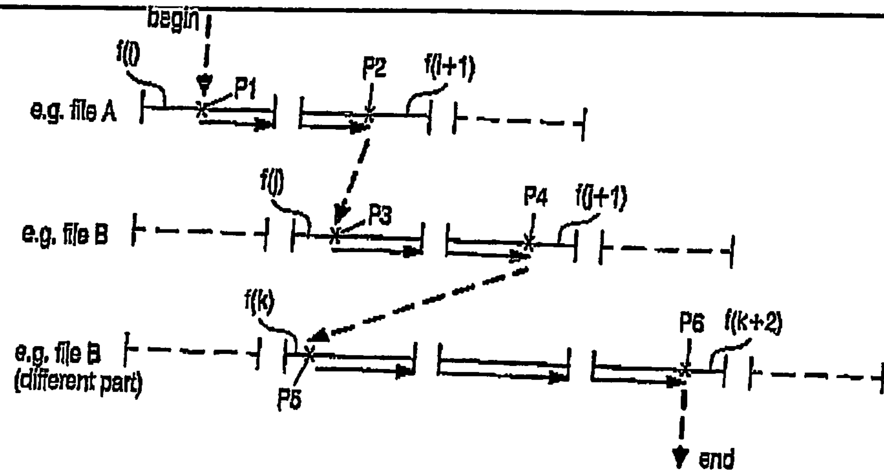

begin
f(i)
P1
P2
f(i+1)
e.g. file A
f(j)
P3
P4
f(j+1)
e.g. file B
f(k)
P6
f(k+2)
e.g. file B
(different part)
P5
end

FIG. 4

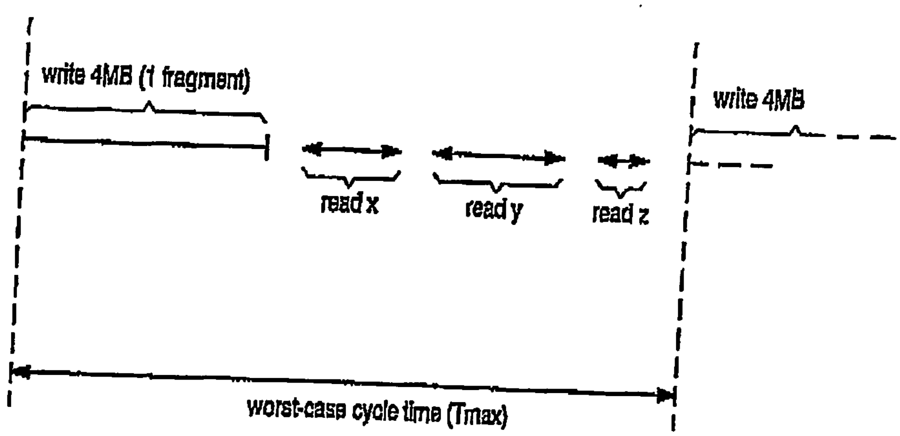

write 4MB (1 fragment)
write 4MB
read x
read y
read z
worst-case cycle time (Tmax)

FIG. 5

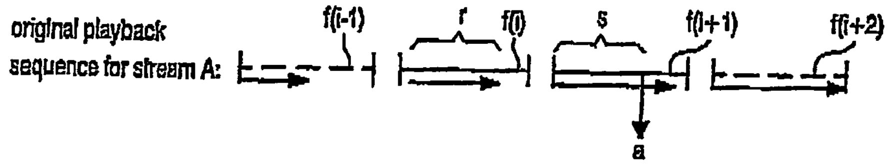


FIG. 6a

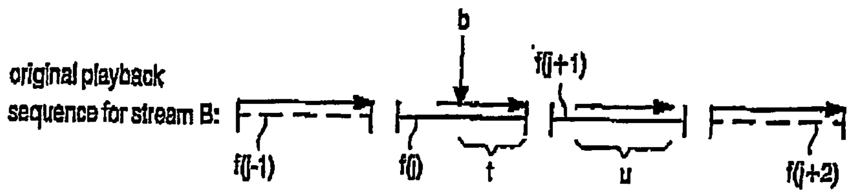


FIG. 6b

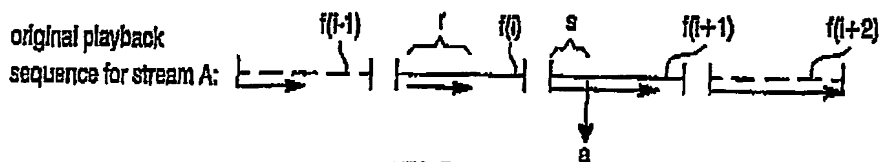


FIG. 7a

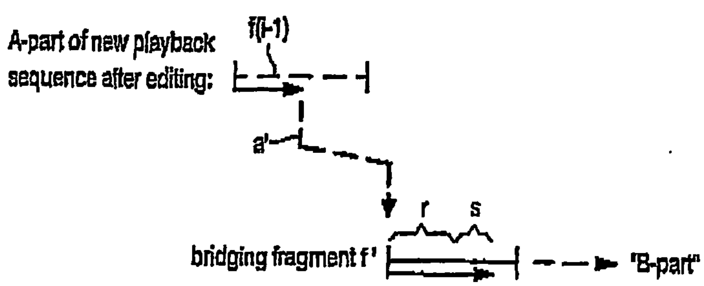


FIG. 7b

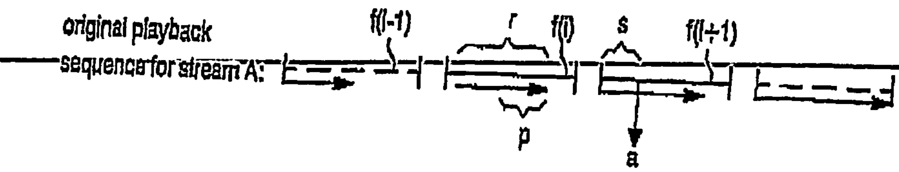


FIG. 8a

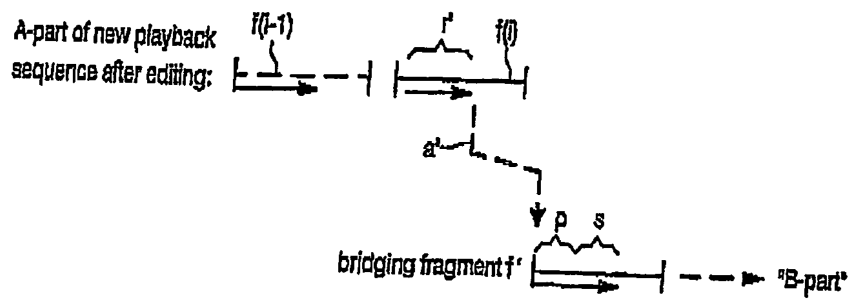


FIG. 8b

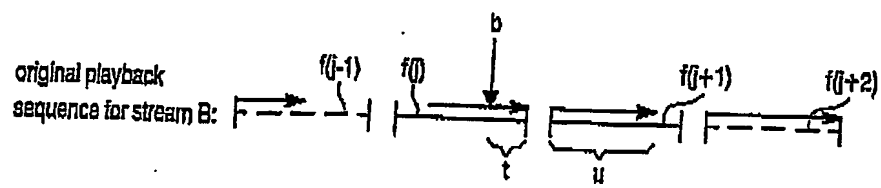


FIG. 9a

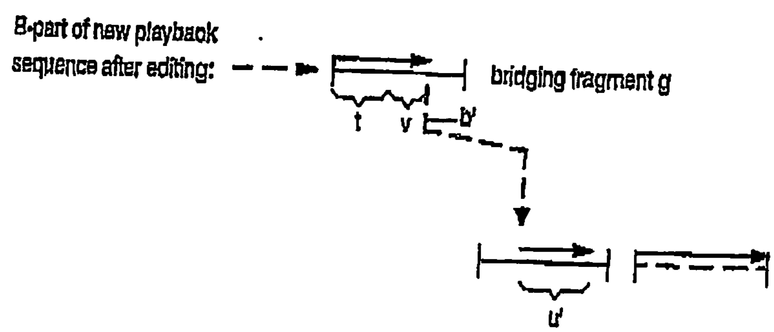


FIG. 9b

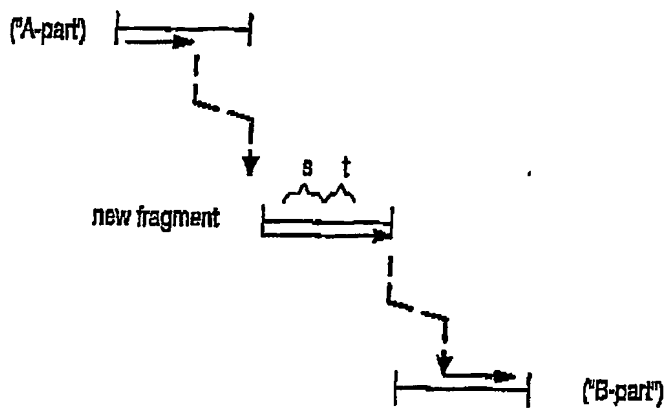

("A-part")
s
t
new fragment
("B-part")

FIG. 10

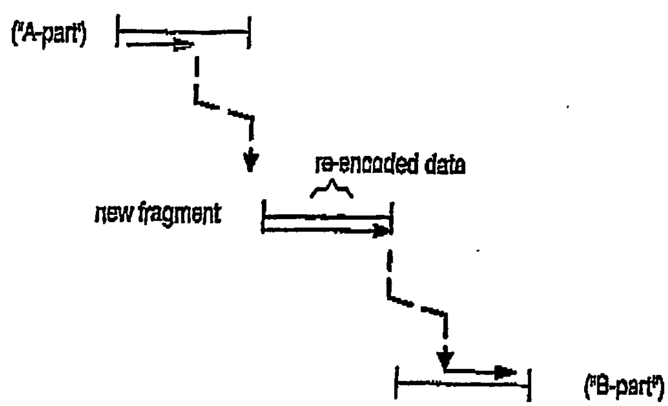

("A-part")
re-encoded data
new fragment
("B-part")

FIG. 11

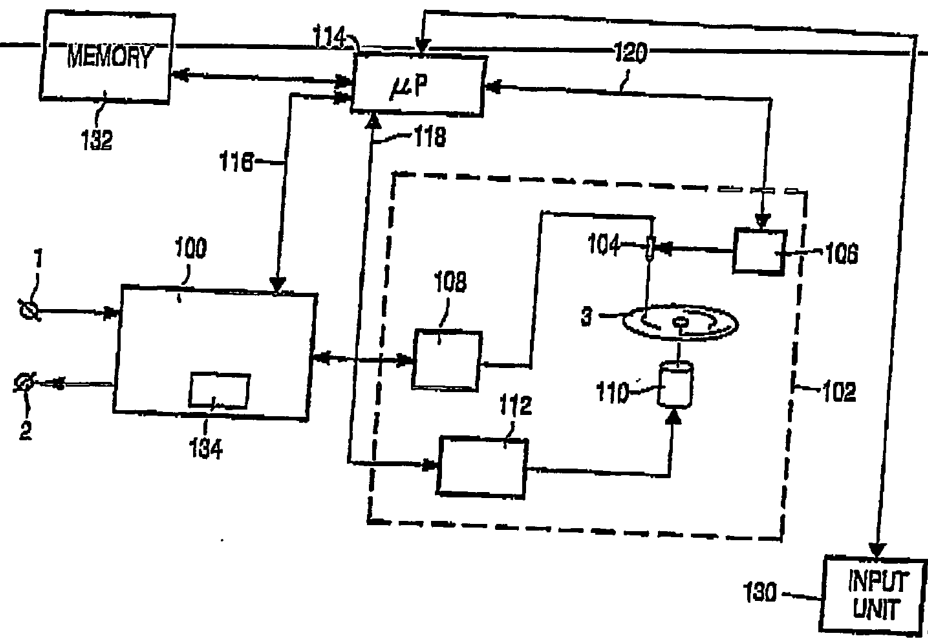

MEMORY
132
114
µP
120
118
116
1
100
108
104
106
3
110
102
2
134
112
130
INPUT UNIT

FIG. 12

ınıS PAGE BLANK (USPTO)